uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería de Sistemas Audiovisuales
2017-2018

Trabajo Fin de Grado

“Diseño y desarrollo de una app basada en Android y Google Spreadsheets”

Autora: Alba Moirón Alén

Tutora: Celeste Campo Vázquez

Leganés, Marzo 2018

AGRADECIMIENTOS

A mi tutora, Celeste, por guiarme en este largo camino.

A mis padres, mi mayor ejemplo a seguir en la vida. Gracias por vuestra paciencia infinita, por vuestro apoyo en los momentos más duros y por confiar siempre en mí, aunque no siempre lo mereciese. Y, sobre todo, gracias por enseñarme que sin esfuerzo y trabajo no se consigue nada en esta vida, y que si quieres algo de verdad tienes que luchar por ello. Siempre llevaré conmigo todos los valores que me habéis transmitido y no puedo dejar de daros las gracias por ello.

A Bea, que me acompaña en todos y cada uno de los momentos de mi vida, me entiende mejor que nadie y me ha soportado y apoyado cuando creía que no podía más y quería tirar la toalla. Gracias por ser mi mejor amiga y estar siempre a mi lado.

A Rocío, por todo su apoyo estos últimos meses. Me has demostrado mucho y nunca lo olvidaré.

A la gente que he conocido durante estos años en la carrera. Todos me habéis aportado algo importante y me habéis hecho crecer un poquito más. Por las largas horas de estudio, los momentos de pánico y las merecidas recompensas.

RESUMEN

El uso de aplicaciones móviles se ha convertido en tendencia en la sociedad en la que vivimos. Cada vez más usuarios dejan de lado sus ordenadores para pasar a usar de forma exclusiva sus smartphones. Por otra parte, el acceso a Google es algo muy extendido entre la población, siendo muy común disponer de una cuenta de correo electrónico de esta compañía, lo que implica la posibilidad de poder acceder a múltiples servicios, como puede ser el paquete ofimático, con aplicaciones online entre las que destaca Google Sheets, una aplicación de hojas de cálculo muy similar a la conocida Excel de Windows.

Es por ello que en este proyecto se ha decidido desarrollar una aplicación móvil basada en Android y en las APIs de Google, con el fin de poder ayudar a la digitalización de pequeños comercios. En muchas ocasiones, pequeñas empresas particulares como librerías, tiendas de ropa o papelerías, sienten la necesidad de consultar y modificar su stock en tiempo real, pero no disponen de una infraestructura informática en red con la cual poder acceder al mismo, lo que lleva a utilizar métodos más rudimentarios, como puede ser un libro en el que se va actualizando la información a mano, lo que puede dar lugar a errores o resultar un trabajo más tedioso y difícil de mantener.

Debido a que la compra y mantenimiento de una infraestructura informática en red supone un coste elevado, que quizás muchas de estas empresas no se puedan permitir, con este proyecto se pretende brindar una alternativa sencilla, intuitiva y barata (o incluso gratuita) al uso de este tipo de infraestructuras, y ofrecer la posibilidad de poder almacenar la información o stock de una pequeña empresa en una hoja de cálculo de Google Sheets, de forma que los propietarios y/o empleados del negocio puedan acceder a la misma, y modificar y consultar la información en tiempo real. De esta manera, en el momento que vendan o reciban productos, podrán reflejarlo en esta “base de datos”, almacenada en los servidores de Google.

En el presente proyecto se ha desarrollado una aplicación de gestión de libros, como ejemplo del uso que se le puede dar a la tecnología de Google Sheets. Se trata de una aplicación cliente-servidor, donde el cliente es Android, y el servidor está basado en Google Sheets y las APIs que proporciona Google. El usuario que utilice esta aplicación puede buscar libros y almacenar su información (título, autor, fecha de publicación, etc.) en su hoja de cálculo personal, así como consultar los mismos cuando lo desee, tanto desde su móvil como desde su ordenador, ya que esta hoja se queda almacenada en el Google Drive asociado a la cuenta proporcionada por el usuario.

ABSTRACT

The use of mobile apps has become a trend in our society. Every day more users leave behind their computers to use their smartphones exclusively. On the other hand, access to Google is widely extended among population, being very common to own an email account of the company. This implies the possibility to access multiple services, like the office package and its applications. Among them is Google Sheets, a spreadsheets application very similar to Windows' Excel.

For this reason, this project is focused on the development of an Android and Google APIs' based app, with the goal of helping small businesses' digitalization. Many times, small particular retailers like libraries, clothes shops or stationaries, feel the need to consult or modify their stock information in real time, but do not have a network structure that allows them to access it. This leads to the use of rudimentary methods, such the update of the information by hand, which implies an added difficulty and time to the job, as well as the risk of making mistakes.

The acquisition and maintaining of a network structure implies an excessive cost, that many of these small businesses might not be able to afford. This project intends to offer a simple and intuitive alternative to the use of network structures, and give the possibility to collect the stock information of a small business in a Google Sheets spreadsheet. This way, the owners and/or employees of the company can access, modify and consult this data in real time. Hence, every time they make a sale or receive new products, they will be able to reflect it in this database that will be stored in Google's servers.

During this project, a book management app has been developed as an example of the use that Google Sheets' technology could be given. It is a client-server app, where the client is Android and the server is based in Google Sheets and the given Google's APIs. The person who uses this app can search books and store their information (title, author, publication date, etc.) on his personal spreadsheet, as well as consulting it when necessary. This could be done both from the smartphone and the computer, as the spreadsheet stays uploaded on the associated Google Drive account given by the user.

ÍNDICE DE CONTENIDOS

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

ACRÓNIMOS

ART	Android Runtime
APP	Aplicación
API	Application Programming Interface *(Interfaz de Programación de Aplicaciones)*
EAN	European Article Number (*Número de Artículo Europeo*)
HAL	Hardware Abstraction Layer *(Capa de Abstracción de Hardware)*
IU	Interfaz de Usuario
JSON	JavaScript Object Notation
PYME	Pequeña Y Mediana Empresa
SHA-1	Secure Hash Algorithm *(Algoritmo Seguro Hash)*
SO	Sistema Operativo

Capítulo 1.

Introducción y objetivos.

En este primer capítulo se presentan las ideas principales del Trabajo Fin de Grado, así como los objetivos que deben cumplirse durante el desarrollo del mismo. Se incluyen también las fases de desarrollo y la planificación llevaba a cabo, y cómo se ha estructurado la presente memoria.

1.1. Introducción

En muchas ocasiones, pequeñas empresas particulares como librerías, tiendas de ropa o papelerías, sienten la necesidad de consultar y modificar su stock en tiempo real, pero no disponen de un servidor con el cual poder acceder al mismo, lo que lleva a utilizar métodos más rudimentarios, como puede ser un libro en el que se va actualizando la información a mano, lo que puede dar lugar a errores o resultar un trabajo más tedioso y difícil de mantener.

Adquirir y mantener un servidor implica un gasto importante de dinero, algo en lo que muchas de estas pequeñas empresas no están dispuestas a invertir o no se pueden permitir, ya que podría no salirles rentable frente a su volumen de ventas. Además del coste que implica la compra del servidor, puede que también se vean en la necesidad de contratar a una persona especializada en esta materia, lo que supone un coste extra.

Hoy en día, el acceso a Google es algo muy extendido en la sociedad, tanto que, a mediados del año 2017, más de 1.2 billones [1] de personas en el mundo disponían de una cuenta de correo electrónico (Gmail) de esta compañía. Tener una cuenta de Google implica la posibilidad de acceder a todos los servicios que ofrece la misma, como puede ser el paquete ofimático, con aplicaciones online entre las que destaca Google Sheets, una aplicación de hojas de cálculo muy similar a la conocida Excel de Windows.

Por ello, con este proyecto se pretende brindar una alternativa sencilla e intuitiva al uso de servidores, ya sean propios o específicos, y ofrecer la posibilidad de poder almacenar la información o stock de una pequeña empresa en una hoja de cálculo de Google Sheets, de forma que los propietarios y/o empleados del negocio puedan acceder a la misma, y modificar y consultar la información en tiempo real. De esta manera, en el momento que vendan o reciban productos, podrán reflejarlo en esta "base de datos", almacenada en los servidores de Google.

Una gran ventaja de este servicio es que es gratuito, en caso de utilizar una cuenta regular de Google, la cual ofrece 15GB por defecto, aunque existe la posibilidad de aumentar el espacio de almacenamiento pagando una pequeña cuota mensual. Google también ofrece la opción de adquirir GSuite [2], un conjunto de aplicaciones enfocadas a aumentar la productividad de las empresas, entre las que están Gmail, Hangouts, Documentos y Sheets, y donde el paquete profesional para oficinas más básico tiene un precio de aproximadamente cuatro euros al mes, lo cual supone un coste mucho menor que la adquisición de un servidor y su mantenimiento.

En este proyecto se ha realizado una aplicación (app) sobre gestión de libros, como ejemplo del uso que se le puede dar a la tecnología de Google Sheets. Se trata de una aplicación cliente-servidor, donde el cliente es Android, y el servidor está basado en Google Sheets y las APIs que proporciona Google. El usuario que utilice esta aplicación puede buscar libros y almacenar su información (título, autor, fecha de publicación, etc.) en su hoja de cálculo personal, así como consultar los mismos cuando lo desee, tanto desde su móvil como desde su ordenador, ya que esta hoja se queda almacenada en el Google Drive asociado a la cuenta proporcionada por el usuario.

1.2. Objetivos

El objetivo principal de este proyecto es estudiar la viabilidad de utilizar Google Sheets como backend desde una aplicación Android. Este proyecto se ha desarrollado, como se ha comentado anteriormente, con el fin de que pequeñas empresas empleen aplicaciones similares a la de este proyecto para almacenar, por ejemplo, información relativa a su stock, sin necesidad de comprar y mantener un servidor.

Este objetivo principal puede dividirse en los siguientes puntos:

- La aplicación debe permitir la creación de una nueva hoja de cálculo de Google Sheets cuando el usuario la instala por primera vez en su smartphone.
- La aplicación debe permitir escribir en la hoja de cálculo, así como consultar la información que hay en la misma.
- La aplicación debe permitir al usuario escanear el código de barras de un libro, con el fin de obtener información relativa al mismo.
- La interfaz de la aplicación debe ser intuitiva y sencilla, para que cualquier usuario pueda entender rápidamente el funcionamiento de la misma.

1.3. Fases desarrollo y planificación

En este apartado se describen las diferentes fases de desarrollo del proyecto, así como la planificación llevada a cabo al inicio del mismo.

1.3.1. Fases desarrollo

- **Fase 1: Planteamiento del proyecto y descarga entorno desarrollo.**

  En esta primera fase se ha expuesto la idea del proyecto, concretando los requisitos e ideas generales del mismo. A su vez, se ha descargado e instalado el entorno de desarrollo, en un principio Android Studio 2.2.3, actualizándose posteriormente a Android Studio 3.1.

- **Fase 2: Documentación y pruebas sobre aplicación de ejemplo.**

  En esta etapa se ha realizado un estudio de las tecnologías empleadas, así como de la aplicación de ejemplo proporcionada por Google (*Google Sheets Quickstart* [3]) para entender su funcionamiento y conocer las funcionalidades que ofrece su API.

- **Fase 3: Programación de la aplicación y primer diseño de la interfaz.**

  En esta fase, la más importante y extensa del proyecto, se comenzó trabajando con las opciones de lectura y escritura desde una hoja de Google Sheets concreta, y una vez que

esto estaba en funcionamiento, se desarrolló la funcionalidad de que la aplicación crease una nueva hoja al instalarse la aplicación, y se comenzó a trabajar con la API de Google Books. También se realizó un primer diseño de la interfaz, buscando una funcionalidad intuitiva y sencilla.

- **Fase 4: Diseño de la interfaz.**

  Aunque en la fase anterior se había hecho un primer esbozo de la interfaz, una vez que la funcionalidad estaba desarrollada, se comenzó a mejorar la interfaz de la aplicación, de modo que fuese más funcional y llamativa para el usuario.

- **Fase 5: Pruebas y control de errores**

  En esta etapa se ha testeado el software para comprobar si realmente funcionaba según lo esperado y se ha llevado a cabo un control exhaustivo de todo tipo de errores.

- **Fase 6: Conclusiones y líneas futuras.**

  En esta fase se extrajeron las conclusiones del proyecto y se analizaron las posibles futuras líneas de trabajo, en las cuales se podría aplicar lo estudiado en el mismo..

- **Fase 6: Redacción de la memoria final.**

  En el momento en el que se finalizaron todas las etapas anteriores, se procedió a la escritura del documento final del proyecto, es decir, la presente memoria, donde se recoge todo lo estudiado y desarrollado durante el transcurso del mismo.

1.3.2. Planificación inicial. Diagrama de Gantt.

A continuación se muestra una tabla con la planificación inicial y el Diagrama de Gantt, donde se recogen los tiempos previstos al inicio del proyecto para cada fase del mismo:

FASE	FECHA INICIO	DURACIÓN (días)	FECHA FIN
Planteamiento del proyecto y descarga entorno desarrollo	01/03/2017	7	08/03/2017
Documentación y pruebas sobre aplicación ejemplo	10/03/2017	10	20/03/2017
Programación de la aplicación	01/04/2017	91	01/07/2017
Diseño final de la interfaz	02/07/2017	44	15/08/2017
Pruebas y control de errores	01/08/2017	15	16/08/2017
Conclusiones y estudio de futuras líneas de trabajo	17/08/2017	4	21/08/2017
Redacción de la memoria final	21/08/2017	61	21/10/2017
	Total (días)	**232**	

Tabla 1. Planificación inicial Trabajo Fin de Grado.

El Diagrama de Gantt correspondiente a la planificación inicial es el siguiente:

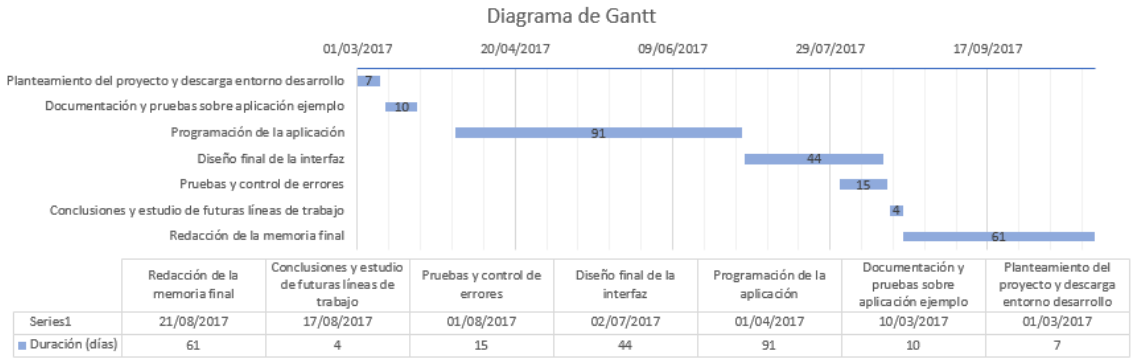

Ilustración 1. Diagrama de Gantt – Planificación inicial.

Es necesario aclarar que esta planificación se corresponde con lo planteado en un primer momento. Debido a la necesidad de compaginar trabajo y estudios se ha producido un retraso temporal de varios meses, hasta llegar a febrero de 2018. No obstante, el tiempo dedicado al proyecto ha sido el mismo que el planificado.

1.4. Estructura del documento

A continuación, se detalla la estructura del presente documento y el contenido de cada uno de sus capítulos:

- **Capítulo 1. Introducción y objetivos.**

  Este capítulo incluye los objetivos y motivaciones del proyecto, la planificación y fases de desarrollo que se han seguido para el mismo, así como la estructura del documento.

- **Capítulo 2. Estado del arte.**

  Este segundo capítulo está dedicado al estudio del estado del arte, donde se introducen y analizan las tecnologías empleadas y sus conceptos básicos, de forma que un lector no especializado en las mismas pueda entender lo básico de este proyecto.

- **Capítulo 3. Descripción de la aplicación.**

  Este capítulo recoge la descripción de la aplicación, mostrando las pantallas de las que se compone y su funcionalidad.

- **Capítulo 4. Diseño e implementación de la aplicación.**

  Este apartado incluye, de una forma más técnica y extensa, información sobre la aplicación y su código. Se muestra el diseño de la arquitectura, tanto de la parte cliente Android, como de la parte servidor, Google Sheets. En cuanto esta última tecnología, se hace un análisis en profundidad sobre la API usada (introducida previamente en el Capítulo 2).

- **Capítulo 5. Pruebas y validación.**

  En este capítulo se recogen las diferentes pruebas realizadas en la aplicación, para validar que el proyecto cumple con los objetivos previstos.

- **Capítulo 6. Conclusiones y líneas futuras.**

  En este capítulo se incluyen las conclusiones sobre el proyecto, y se describen las líneas de trabajo con las que se podría seguir trabajando en un futuro.

- **Capítulo 7. Entorno socioeconómico, presupuesto y margo regulador.**

  En el capítulo final se analiza el entorno socioeconómico y el posible impacto que podría tener este proyecto en el mismo, incluyendo un presupuesto final de desarrollo de la aplicación. Se estudia también el marco regulador, centrándose concretamente en la *Ley de Protección de Datos de Carácter Personal,* para su posterior aplicación, y en la *Política de Privacidad de Google*.

Capítulo 2.

Estado del arte.

En este capítulo se presentan los conceptos principales sobre las tecnologías utilizadas en el proyecto, así como una breve introducción a las mismas en cuanto a su arquitectura y/o funcionamiento. Adicionalmente se incluye un breve estudio de aplicaciones similares existentes en el mercado, con el fin de recolectar ideas sobre lo que una aplicación de este tipo debería incluir.

2.1. Tecnología cliente-servidor

Hoy en día, la mayoría de los servicios se componen de dos partes, cliente y servidor. En la mayoría de los casos, la primera ofrece una versión móvil, ya que hay un gran número de usuarios, el cual crece cada día, que acceden a los servicios solamente desde sus dispositivos móviles. Esta versión cliente se desarrolla normalmente para los sistemas operativos dominantes en el mercado en la actualidad, es decir, Android e iOs.

En cuanto a la parte de servidor, en este proyecto se ha elegido trabajar con Google Sheets, Google Books y sus respectivas APIs.

A continuación, se explicará más en profundidad la tecnología cliente-servidor de este proyecto.

2.1.1. Cliente

En este apartado se hará una breve introducción a Android y su historia, y se mostrarán los conceptos básicos de este sistema operativo (SO).

2.1.1.1. Android

Android es un sistema operativo basado en Linux, con un uso orientado principalmente a dispositivos móviles, como smartphones y tablets. Hoy en día se comienza a extender su uso en smartwatches, televisores e incluso en automóviles.

Se trata de un SO gratuito y de código abierto. Para desarrollar en Android, solamente se necesita conocimiento sobre Java y la descarga del entorno de programación Android Studio, el cual está disponible para los principales sistemas operativos (Windows, Ubuntu, macOS, ...).

En el año 2005, cuando Android Inc. contaba con solamente 22 meses de vida, Google compra la compañía, y posteriormente, en el año 2007, se crea la OAH (Open Handset Alliance), un consorcio de 47 compañías de hardware, software y telecomunicaciones, con el objetivo principal de desarrollar estándares abiertos para dispositivos móviles, y cuyo producto principal era la plataforma Android. Hoy en día la OAH está formada por un total de 84 compañías del sector tecnológico y de las telecomunicaciones.

Desde sus inicios, el sistema operativo ha tenido numerosas actualizaciones, las cuales, a medida que iban saliendo a la luz, arreglaban *bugs* y añadían nuevas funcionalidades respecto a las anteriores. La primera versión, Android 1.0: Apple Pie, se lanzó en septiembre de 2008, llegando hasta la última versión, Android 8.0: Oreo, estrenada en agosto de 2017. Como dato curioso, todas las versiones se bautizaron con el nombre de un dulce, por orden alfabético.

2.1.1.1.1. Conceptos básicos

Como se ha comentado en el punto anterior, Android ofrece una plataforma abierta de desarrollo y un sistema operativo con núcleo Linux, el cual también es de código abierto.

La imagen que se muestra a continuación representa la arquitectura del SO:

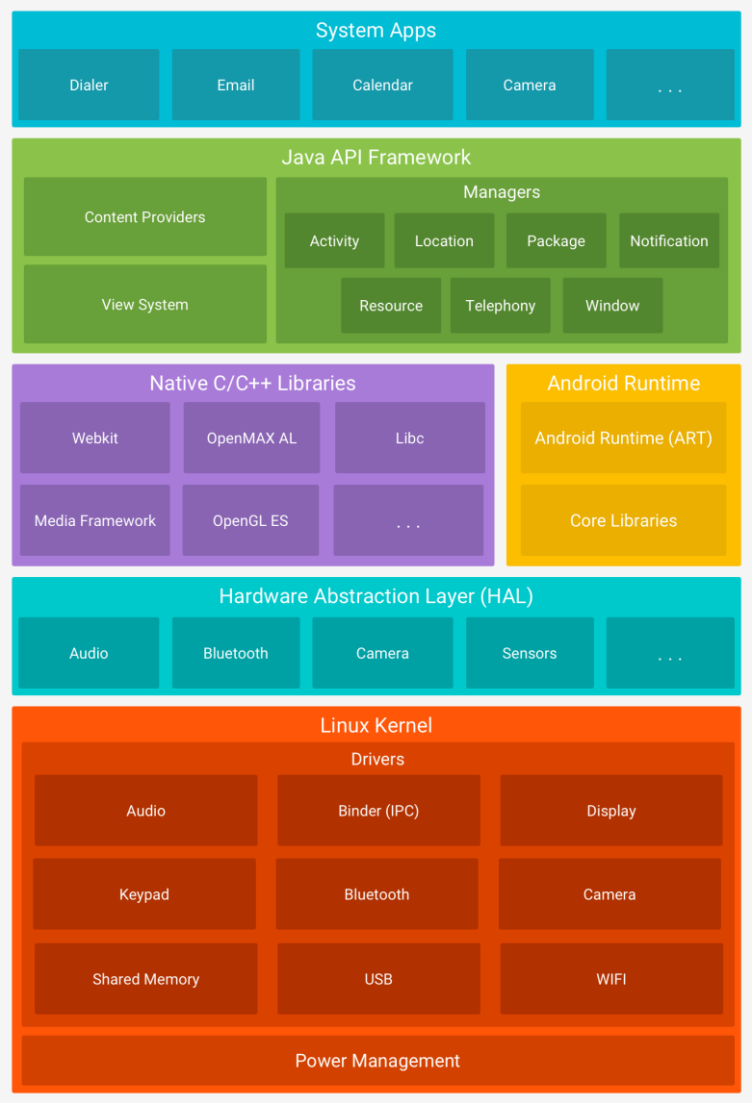

Ilustración 2. Arquitectura Android. Fuente [4].

Analizando cada una de las partes que componen la arquitectura del sistema:

- **Aplicaciones del sistema**. El sistema operativo incluye por defecto un conjunto de aplicaciones como correo electrónico, calendario, contactos, mensajería SMS (entre otras), para brindar al usuario la oportunidad de utilizar las mismas en el desarrollo de su propia aplicación. Sin embargo, la mayoría de estas aplicaciones pueden ser sustituidas por otras en caso de que el usuario lo desee.

- **Java API Framework**. Esta capa de la arquitectura se diseñó con el fin de simplificar la reutilización de componentes, y proporciona las bases para crear aplicaciones Android mediante APIs escritas en Java, es decir, los desarrolladores pueden acceder a las mismas APIs que usan las apps que vienen por defecto en el sistema Android. Dentro de esta capa, los componentes más destacados son:
    - **View System**: conjunto de vistas para poder compilar la IU (Interfaz de Usuario). Se compone de listas, cuadrículas, botones y cuadros de texto, entro otros.
    - **Activity Manager**: controla el ciclo de vida de las apps y proporciona una pila de navegación común a todas ellas.
    - **Notification Manager**: permite que las apps muestren alertas personalizadas en la barra de estado.
    - **Resource Manager**: proporciona acceso a recursos sin código, como gráficos o archivos de diseño.
    - **Content Providers**: los proveedores de contenido proporcionan un sencillo mecanismo para que se pueda acceder a los datos de una aplicación desde otra.
- **Librerías nativas C/C++.** Estas librerías están compiladas en código nativo del sistema. La decisión de que estén escritas en C/C++ se tomó debido a que proporcionan mayor rendimiento y rapidez, y menor consumo de memoria que Java. Como se puede ver en la imagen de arriba, algunas de estas librerías son Webkit, OpenGL ES, Media Framework, OpenMAX AL y Libc, entre otras.
- **Android Runtime (ART).** El tiempo de ejecución ART apareció con la versión 5.0 de Android (previamente el tiempo de ejecución del sistema operativo era Dalvik). Se basa en el concepto de máquina virtual utilizado en Java, está optimizado para ocupar un espacio de memoria mínimo, y en esta nueva versión se consigue reducir el tiempo de ejecución del código Java hasta en un 33%. También proporciona un conjunto de bibliotecas básicas de Java y específicas de Android.
- **Hardware Abstract Layer (HAL).** La capa de abstracción de hardware está formada por varios módulos de biblioteca. Cada uno de estos módulos implementa una interfaz para un tipo concreto de componente de hardware, como puede ser el módulo de bluetooth o el de cámara.
- **Linux Kernel.** El núcleo de Linux 2.6 es la base del sistema Android, y actúa como capa de abstracción entre el hardware y el resto de la pila, lo que implica que es dependiente del hardware. Esta última capa proporciona servicios como el manejo de la memoria, la seguridad y el soporte de drivers para dispositivos, entre otros.

Por otra parte, cabe destacar que el desarrollo en Android está orientado a componentes, por lo que una aplicación está formada por uno o varios de éstos. A continuación se introducen los principales componentes de las aplicaciones Android:

- **Activity:** normalmente representa una pantalla con interfaz de usuario. Se implementa como subclase de la clase `Activity`. Toda aplicación Android debe tener como mínimo un componente de este tipo.

- **Services:** componente que modela una actividad que se ejecuta en segundo plano, normalmente debido a su carga computacional o a la necesidad de ejecutar una tarea mientras se realiza otra. Este componente no tiene asociada una interfaz gráfica de usuario.

- **Broadcast Receivers:** se utiliza para recibir y responder ante anuncios de mensajes enviados por difusión. No realiza ninguna acción, solo reacciona a un anuncio lanzando alguna actividad que procese la información. La mayoría de los mensajes son originados por el sistema, como, por ejemplo, un aviso de que la batería se está agotando. Las aplicaciones también pueden generar sus propios mensajes y difundirlos para dar aviso a otras aplicaciones.

- **Content Providers:** a través de los proveedores de contenido, otras aplicaciones pueden consultar o modificar, en caso de que lo permitan, los datos. Estos datos pueden estar almacenados en una base de datos, en el propio sistema o en cualquier otro tipo de almacenamiento. En resumen, su función es proporcionar un conjunto de datos de una aplicación para que estén disponibles para otras.

Activities, Services y Broadcast Receivers se activan mediante `Intents`, el elemento básico de comunicación entre los distintos componentes. Los `Intents` permiten arrancar de forma explícita una `Activity` o `Service` indicando su nombre de clase, declarar de forma implícita la intención de que una `Activity` o `Service` se arranque para realizar una acción, y difundir que ha ocurrido un evento o una acción para el caso de los Broadcast Receivers.

Todas las aplicaciones Android son resultado de una combinación de uno o más de estos componentes. Todos deben ser declarados de forma explícita en el manifiesto (fichero *AndroidManifest.xml*), cuya función es informar al sistema acerca de dichos componentes de la aplicación, además de declarar la mínima versión de Android en la que puede funcionar la app (lo que se conoce como nivel de API mínimo), identificar los permisos de usuario que requiere la misma, o incluir las características de hardware que la aplicación necesita (como puede ser la cámara, la localización o el bluetooth).

2.1.1.1.2. Elección del sistema operativo

Además del sistema operativo Android, el cual se ha explicado previamente, otro de los SO dominantes en el mercado es iOs, el sistema operativo creado por la compañía Apple Inc., y cuya primera versión salió a la luz en el año 2007, acompañado del revolucionario iPhone. A diferencia

de Android, iOS solo puede usarse en dispositivos móviles de esta compañía y, además, las herramientas de desarrollo sólo están disponibles en ordenadores MAC.

Haciendo una comparativa de estos dos sistemas operativos, sin duda alguna el que más destaca en el mercado hoy en día es Android. Como se puede comprobar en la gráfica que se adjunta a continuación, hasta mayo de 2017, Android dominaba el 85% del mercado de smartphones, mientras que iOS solo ocupada el 14,7%, seguido por Windows Phone con un 0.1 %.

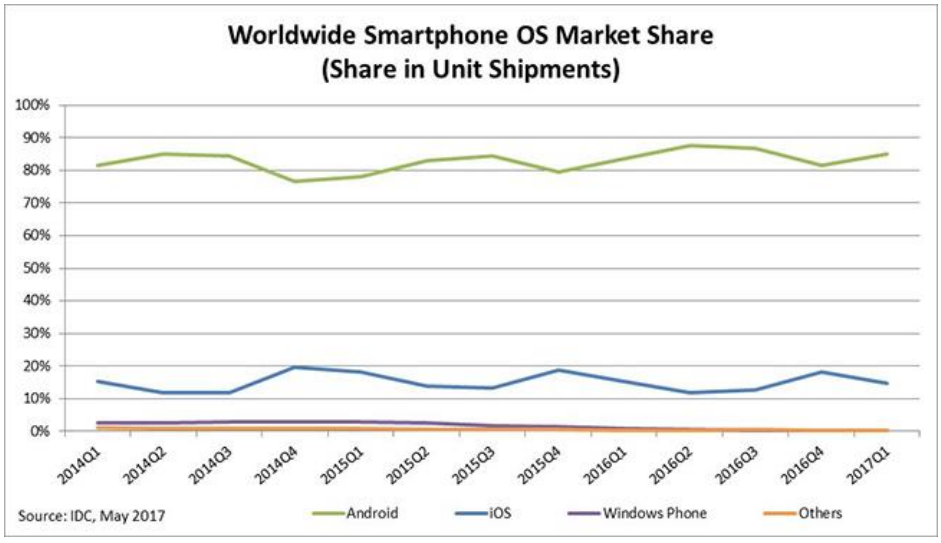

Ilustración 3. Uso de los principales sistemas operativos. Fuente [5].

Por este motivo se ha decidido crear una aplicación en Android, pensando en que, en un futuro, sería el sistema operativo desde el cuál un mayor número de personas podría acceder a la aplicación desarrollada. Además de esto es lógico pensar que, trabajando con tecnología de Google, empresa a la cual pertenece este SO, la elección del mismo es la más sensata. Sin embargo, es necesario aclarar que también podría desarrollarse bajo iOs, puesto que las APIs de servidor se pueden utilizar desde este sistema operativo e incluyen documentación para el mismo.

En cuanto a la versión de Android, se ha decidido desarrollar la aplicación de este proyecto sobre la versión 7.0, puesto que, por detrás de la última versión (Android 8.0: Oreo), es la más reciente y una de las más usadas hoy en día. La mayoría de smartphones que se venden en el presente ya incluyen esta versión, y como se puede ver en el gráfico que se muestra a continuación, es una de las más usadas. Es importante destacar que las diferentes versiones de Android garantizan compatibilidad hacia delante. Por ejemplo, para este caso concreto, además de funcionar sobre la versión 7.0, en principio también podrá usarse en la 8.0 y posteriores actualizaciones.

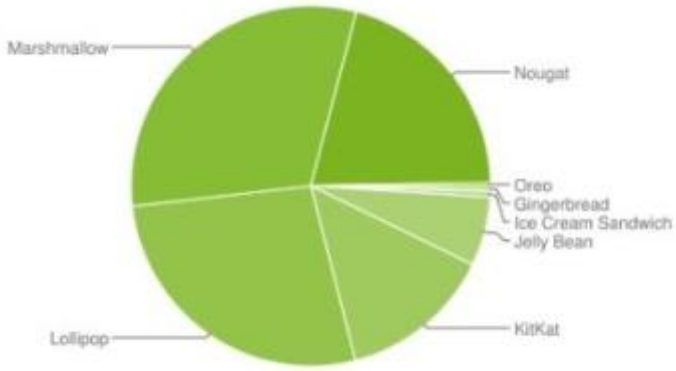

Ilustración 4. Uso de versiones Android hasta noviembre 2017. Fuente [6].

2.1.2. Servidor

En este apartado se hará una introducción a Google Sheets, Google Books y sus respectivas APIs, y se mostrarán los conceptos básicos de las mismas.

2.1.2.1. Google Sheets

Google Sheets, disponible desde junio de 2006, es una aplicación gratuita de hojas de cálculo en línea, que permite a sus usuarios crear y modificar las mismas, darles formato y trabajar simultáneamente con otras personas. Se puede acceder mediante un ordenador, teléfono móvil o tablet, teniendo incluso la opción de consultar una hoja de cálculo sin necesidad de conexión a Internet. Incluye un chat para que varios usuarios puedan hablar mientras modifican en tiempo real una hoja de cálculo. Los cambios se guardan automáticamente mientras se realizan modificaciones, y se puede consultar un historial con versiones anteriores y cambios recientes. Además, la aplicación permite abrir, editar y guardar archivos en formato Excel.

Su uso es posible a través de la API de Google Sheets, que permite al código crear y modificar hojas de cálculo. Para poder trabajar con ella, solamente se requiere tener una cuenta Google y aceptar los permisos requeridos de acceso a contactos y almacenamiento.

2.1.2.1.1. API Google Sheets

La API de Google Sheets [7], la cual se encuentra en su cuarta versión, permite leer y modificar cualquier aspecto de las hojas de cálculo. Antes de profundizar en esto, y como introducción a la API, se muestran los términos comunes que un usuario se puede encontrar en la misma. Es necesario aclarar que, cuando se habla de una hoja de cálculo (*spreadsheet*) en este documento, en realidad se está haciendo referencia a un conjunto de hojas (*sheets*) de cálculo, es decir, un libro, donde se pueden crear tantas hojas como desee el usuario. Se utilizará esta nomenclatura ya que es la proporcionada por la API de Google.

- **ID la hoja de cálculo (*spreadsheet*)**: Todos los métodos que se incluyen en la API requieren el parámetro `spreadsheetId`, el cual se utiliza para identificar el conjunto de hojas de cálculo con el que se está trabajando. Este ID, formado por números, letras y algunos caracteres especiales, se encuentra en la URL del conjunto de hojas de cálculo, situado entre */d* y */edit*, como se puede ver a continuación, señalado en negrita:

  `https://docs.google.com/spreadsheets/d/`**`1qpyC0XzvTcKT6EISywvqESX3A0MwQoFDE8p-Bll4hps`**`/edit#gid=0`

- **ID de una hoja concreta (*sheet*)**: En un libro de cálculo, cada una de sus hojas tiene un título (que debe ser único) y un ID. Para especificar la hoja con la que se está trabajando, se utiliza `sheetId`, el cual se encuentra también en la URL de la hoja de cálculo, como el valor del parámetro `gid`. A continuación, se muestra la estructura de la URL y la ubicación de `sheetId`:

  `https://docs.google.com/spreadsheets/d/`*`spreadsheetId`*`/edit#gid=`***`sheetId`***

- **Notación A1:** Algunos métodos de esta API requieren que los rangos estén en notación A1. Se trata de un `String` del tipo `Sheet1!A1:B2`, el cual hace referencia a un grupo de celdas de la hoja de cálculo, y se usa generalmente en fórmulas. A modo de ejemplo, se muestran una serie de ejemplos de rangos válidos para esta notación:

  - `Sheet1!A1:B2`: este rango hace referencia a las dos primeras celdas en las dos primeras filas de la hoja "Sheet1".
  - `Sheet1!A:A`: hace referencia a todas las celdas en la primera columna de la hoja "Sheet1".
  - `Sheet1!1:2`: hace referencia a todas las celdas en las dos primeras filas de la hoja "Sheet1".
  - `Sheet1!A3:A`: hace referencia a todas las celdas de la primera columna de la hoja "Sheet1", a partir de la fila 3.
  - `A1:B2`: hace referencia a las primeras dos celdas de las dos primeras filas de la primera hoja visible.
  - `Sheet1`: hace referencia a todas las celdas de la hoja "Sheet1".

  También se admiten rangos con nombre, dándole siempre prioridad a este en el caso de que entre en conflicto con el nombre de una hoja.

- **Números en serie para fecha/hora**: Como en la mayoría de aplicaciones de hojas de cálculo, en Google Sheets se considera a los valores de fecha y hora como valores decimales, de forma que se pueden realizan operaciones aritméticas con estos valores

mediante fórmulas. Google Sheets utiliza un tipo de fecha `epoch`, usada frecuentemente en hojas de cálculo. La porción de número entero del valor (a la izquierda del decimal) cuenta los días desde el 30 de diciembre de 1899. La porción fraccionaria (a la derecha del decimal) cuenta el tiempo como una fracción de un día.

- **Objeto `ValueRange`:** Se refiere a los datos dentro de un rango de la hoja de cálculo. Dentro de este objeto se pueden tener varios campos:
    - `range`: el rango en el que están los valores de interés, en notación A1.
    - `majorDimension`: la dimensión principal de los valores. Para escritura, si no se especifica, tendrá por defecto la opción "ROWS", es decir, filas. Si se necesita que la dimensión sea columnas, habrá que poner el valor "COLUMNS" en este parámetro.
    - `values[]`: los valores a leer o escribir. Esto será un array de arrays, donde el array exterior representa todos los datos, y cada array interior representa la dimensión principal. Cada elemento del array interior se corresponde con una celda de la hoja.

Los métodos que ofrece el objeto `ValueRange`, y en los cuales se profundizará posteriormente, son los siguientes:

- `append`
- `batchClear`
- `batchClearByDataFilter`
- `batchGet`
- `batchGetByDataFilter`
- `batchUpdate`
- `batchUpdateByDataFilter`
- `clear`
- `get`
- `update`

Como se ha comentado anteriormente, esta API permite leer y modificar cualquier aspecto de las hojas de cálculo. Las hojas de cálculo pueden tener a su vez varias hojas, y cada una de ellas puede tener cualquier cantidad de filas y columnas. Una celda es la intersección de una fila y una columna determinadas, y puede incluir un valor de datos, ya sea numérico o alfanumérico.

La API ofrece dos formas de interactuar con la hoja de cálculo:

- **Leer o escribir sólo valores de celdas**. Esto se hace a través de la colección `spreadsheets.values`. Esta colección ofrece los siguientes métodos:

Acceso a rangos	Lectura	Escritura
Un rango	`spreadsheets.values.get`	`spreadsheets.values.update`
Varios rangos	`spreadsheets.values.batchGet`	`spreadsheets.values.batchUpdate`
Anexos		`spreadsheets.values.append`

Tabla 2. Lectura y escritura de valores de celdas.

- **Lectura**

Para poder leer una hoja de cálculo, es necesario disponer del ID de la hoja de cálculo (`spreadsheetId`) y la notación A1 de los rangos. Existen tres parámetros opcionales que son los encargados de controlar el formato de salida:

Parámetro de formato	Valor predeterminado
`majorDimension`	ROWS
`valueRenderOption`	FORMATTED_VALUE
`dateTimeRenderOption`	SERIAL_NUMBER

Tabla 3. Parámetros lectura API Google Sheets.

El parámetro `dateTimeRenderOption` solo se usa en caso de que `valueRenderOption` no sea FORMATTED_VALUE.

A continuación se muestran los métodos `get` para leer uno o varios rangos de la hoja:

- **Leer un solo rango**: se usa una solicitud `spreadsheets.values.get`, la cual devuelve un objeto `ValueRange`.

- **Leer varios rangos**: para leer varios rangos no contiguos, se usa una solicitud de tipo `spreadsheets.values.batchGet`, que permite especificar cualquier cantidad de rangos que se quieran recuperar. Esta solicitud devuelve un objeto `BatchGetValueResponse`, el cual contiene el ID de la hoja de cálculo y una lista de objetos `ValueRange`.

- **Escritura**

Al igual que en el caso de lectura, para poder escribir en una hoja de cálculo, es necesario disponer del ID del conjunto de hojas de cálculo (`spreadsheetId`) y la notación A1 de los rangos. También se necesitan los datos a escribir ordenados en un objeto de cuerpo de solicitud adecuado. Para las actualizaciones, es necesario el uso de un parámetro `ValueInputOption` válido, el cual controla si los `String` de entrada se analizan o no, tal y como se describe en la siguiente tabla:

ValueInputOption	Descripción
`RAW`	La entrada no se analiza, simplemente se inserta como un `String`. Es decir, los valores que no sean de este tipo, como booleanos o fórmulas, se insertan igualmente como `String`.
`USER_ENTERED`	La entrada se analiza exactamente como si se hubiese ingresado en la IU de Google Sheets. Por ejemplo, si se inserta "Sep 1 2016", se convierte en una fecha

Tabla 4. Parámetros escritura API Google Sheets.

Los métodos existentes para actualizar uno o varios rangos de la hoja son los siguientes:

- **Escribir un solo rango**: se usa la solicitud `spreadsheets.values.update`.
- **Escribir varios rangos**: para leer varios rangos no contiguos, se usa una solicitud de tipo `spreadsheets.values.batchGet`, donde es necesario especificar el `spreadsheetId`, un `valueInputOption` y uno o más `ValueRange`.

- **Leer o escribir cualquier aspecto de la hoja de cálculo**, a través de la colección `spreadsheets`. Además de los datos de sus celdas, una hoja incluye formatos de celda, bordes de celda, rangos con nombre, rangos con protección y formatos adicionales. Estos son algunos de los tipos de datos que controlan la apariencia y el funcionamiento de una hoja de Google Sheets. A través del método `batchUpdate` se puede actualizar cualquiera de estos elementos. Las operaciones específicas que admite este método se pueden agrupar en las siguientes categorías:

Categoría	Descripción
Agregar y duplicar	Agregar nuevos objetos; a veces a partir de antiguos, como en las solicitudes para duplicar
Actualizar (y definir)	Actualizar ciertas propiedades de un objeto sin modificar
Borrar	Quitar objetos

Tabla 5. Lectura y escritura de cualquier aspecto de la hoja de cálculo.

La app desarrollada utiliza Google Sheets como base de datos, de forma que toda la información que el usuario busca en la misma puede almacenarse en una hoja de cálculo a la que puede acceder en cualquier momento para consultar o modificar la información que contiene. La hoja de cálculo del usuario también se mostrará en su Google Drive, pudiendo acceder a ella desde ahí si lo desea. En el Capítulo 4 se profundizará más en el funcionamiento de esta API y el uso que se le ha dado a la misma.

2.1.2.1.2. API Google Books

La API de Google Books [8], la cual se encuentra en su primera versión experimental, ofrece la posibilidad de acceder a información sobre libros, consultar la disponibilidad de eBooks e incluso acceder a textos completos. Permite incluir, en páginas web con JavaScript, un visor para leer libros completos.

Para poder usarla desde una aplicación cliente, es necesario adquirir una clave API e introducirla en el `Manifest` de la aplicación móvil. Para esto, hay que acceder a la Consola de Developers de Google[1]. Solamente es necesario buscar la API y activarla, asociándola al proyecto que incluye la aplicación, y a continuación incluir la clave de API proporcionada en el `Manifest`, de la siguiente forma:

```
<meta-data
    android:name="com.google.android.books.API_KEY"
    android:value = "AIzaSyCpYxz5556X4UzPV6rFxxxxsCs_Q_x"/>
```

Esta API está basada en cuatro conceptos básicos:

- **Volumen**: representa la información que la API de Google Books tiene almacenada sobre un libro o una revista. Es el recurso primario de la API; todos los demás contienen o referencian un volumen.

[1] https://console.developers.google.com/apis/

- **Librería**: una librería es una colección de volúmenes. Google Books ofrece un conjunto de librerías predefinidas para cada usuario, algunas de las cuales están completamente administradas por el mismo, otras se rellenan automáticamente en base a la actividad del usuario, y otras son mixtas. Los usuarios pueden crear, modificar y borrar las librerías, y estas pueden ser públicas o privadas, según se desee.

- **Reseña:** la reseña de un volumen es una combinación de un texto y/o una valoración que se otorga asignando un número de estrellas, siendo el mínimo una estrella, y el máximo cinco. Un usuario puede añadir una reseña por volumen.

- **Posición en la lectura**: indica la última posición en la que se quedó un usuario en un volumen, es decir, la página en la que se quedó en la última lectura. Un usuario solamente puede tener una posición registrada por volumen. Si el usuario no ha abierto dicho volumen todavía, este parámetro no existe. Esta información es privada para el usuario.

Para poder acceder a la información de un libro se ha utilizado JSON (*JavaScript Object Notation*), un formato para el intercambio de datos. Para ello, primero es necesario que el usuario escanee el código de barras de un libro. A través de su código EAN13 [9], la API de Google Books buscará el mismo y devolverá una URL con los datos del libro, con contenido similar a la imagen que se muestra a continuación.

En el código, a través de JSON, se recuperarán aquellos parámetros que se deseen. En este caso, el código busca el título del libro, autor, fecha de publicación, puntuación, sinopsis y categoría.


```
{
 "kind": "books#volumes",
 "totalItems": 860,
 "items": [
  {
   "kind": "books#volume",
   "id": "2EaOj7-ozKgC",
   "etag": "rvitqX80Cs0",
   "selfLink": "https://www.googleapis.com/books/v1/volumes/2EaOj7-ozKgC",
   "volumeInfo": {
    "title": "Harry Potter y el prisionero de Azkaban",
    "authors": [
     "J.K. Rowling"
    ],
    "publisher": "Pottermore",
    "publishedDate": "2015-12-08",
    "description": "De la prisión de Azkaban se ha escapado un terrible villano, Sirius Black, un asesino en serie que fue
vengarse de Harry por haber destruido a su maestro. Por si esto fuera poco, entran en acción los dementores, unos seres abo
eliminar todo recuerdo hermoso de aquellos que se atreven a acercárseles. El desafío es enorme, pero Harry, Ron y Hermione
    "industryIdentifiers": [
     {
      "type": "ISBN_13",
      "identifier": "9781781101339"
     },
     {
      "type": "ISBN_10",
      "identifier": "1781101337"
     }
    ],
    "readingModes": {
     "text": true,
     "image": true
    },
    "pageCount": 360,
    "printType": "BOOK",
    "categories": [
     "Juvenile Fiction"
    ],
    "averageRating": 4.5,
```


Ilustración 5. Ejemplo URL JSON proporcionada por API Google Books.

2.2. Aplicaciones similares en el mercado

En este apartado se analizan aplicaciones de propósito similar a la que se ha desarrollado en el proyecto. Los ejemplos que se darán a continuación tienen que ver con la finalidad de la aplicación que se ha desarrollado, pero es necesario aclarar que ninguna de ellas utiliza Google Sheets como backend, solamente se han utilizado como forma de inspiración en cuanto a funcionalidad de la aplicación desarrollada.

2.2.1. Goodreads

Goodreads [10] es una aplicación de uso exclusivo para teléfonos móviles con SO Android. Se puede obtener de forma gratuita a través de Google Play, el *market* de aplicaciones de Android.

Esta aplicación ofrece un catálogo de más de 12 millones de libros, con los que el usuario puede formar estanterías digitales y listas, con la posibilidad de ordenarlos por categorías o por el estado en el que se encuentra el libro para el lector (leído/no leído/pendiente de lectura). El usuario puede otorgar a cada libro una valoración de entre 1 y 5 estrellas, y también puede escribir una crítica sobre el mismo, con el fin de que otros usuarios puedan leerla.

También es importante destacar que esta app ofrece la posibilidad de conectar con amigos a través de Facebook o de los contactos del propio teléfono. Incluye un apartado de "Recomendaciones" y otro de "Mejores libros del año", y también una opción para escanear códigos de barra de libros, para su posterior almacenamiento en una de las listas.

A continuación se muestra la pantalla principal de la aplicación:

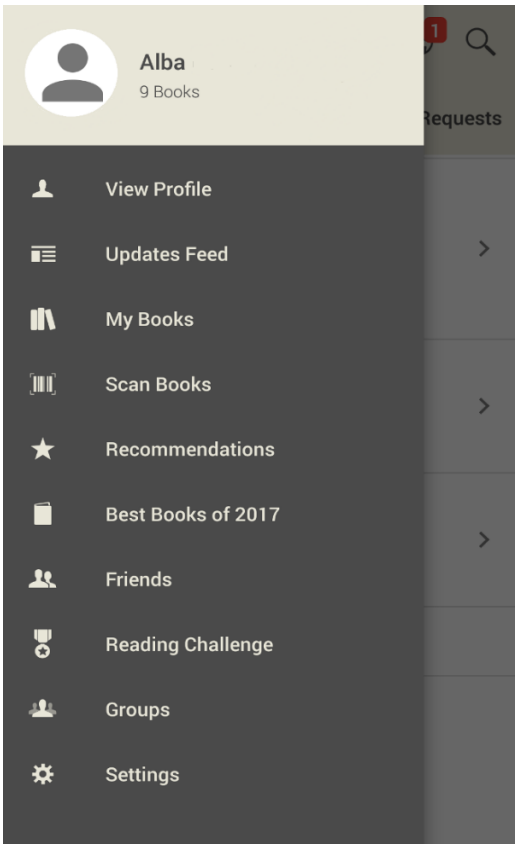

Ilustración 6. Interfaz principal GoodReads.

2.2.2. iReadItNow

Esta aplicación [11] es muy similar a la anterior, pero en este caso está diseñada solamente para iOS, el sistema operativo de Apple, por lo que solo puede descargarse a través del App Store de la compañía. Ofrece también una versión de pago para iPad, llamada iReadItNow HD.

Al igual que GoodReads, permite archivar libros eligiendo su estado (leído, no leído, pendiente de lectura) y escanear códigos de barras para almacenar nuevos libros. Ofrece la opción de sincronizar la información en múltiples dispositivos, y de compartir lo que el usuario está leyendo a través de Facebook y Twitter.

La pantalla principal de la interfaz gráfica de la app es la siguiente:

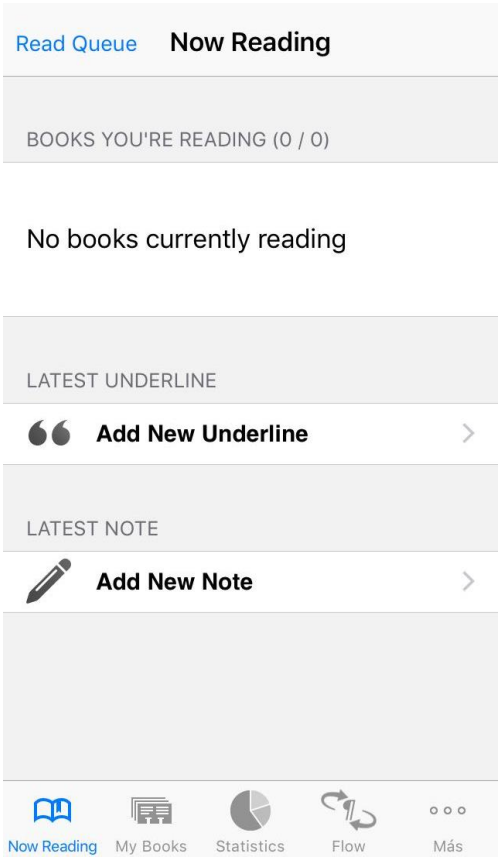

Ilustración 7. Interfaz principal iReadItNow.

2.2.3. Conclusiones

Para el desarrollo de la aplicación de este trabajo, se han tomado como ejemplo algunas de las funcionalidades que incluyen las apps mencionadas anteriormente. Por ejemplo, se ha incluido el escaneo de códigos de barras, y la funcionalidad de mostrar un listado con todos los libros que el usuario tiene almacenados. También se ha tomado la idea de mostrar la valoración de un libro y la función de poder compartir información sobre la aplicación en redes sociales.

Capítulo 3.

Descripción de la aplicación.

En este capítulo se presenta, a modo de manual de usuario, la descripción de la aplicación, incluyendo las diferentes pantallas que la componen, lo que se puede encontrar en cada una, y cómo es la navegación entre ellas. Esta descripción facilitará el seguimiento del Capítulo 4, donde se detalla técnicamente cómo se ha implementado esta aplicación.

3.1. Funcionamiento y pantallas de la aplicación

Cuando se instala la aplicación (a la cual se le ha dado el nombre ***Bookshelf***) en un dispositivo móvil y se arranca, aparece un diálogo en el que se pregunta al usuario si desea crear una nueva hoja de cálculo de Google Sheets, requisito indispensable para utilizar la aplicación. Si el usuario pulsa sobre la opción "No", se mostrará un mensaje informando que es necesario crear la hoja, y se volverá a ofrecer la opción, tal y como se muestra a continuación:

Ilustración 8. Diálogo para crear una nueva hoja de Google Sheets.

Una vez el usuario haya pulsado sobre la opción "Sí" en el diálogo anterior, se pide permiso para poder acceder a sus contactos. Si el usuario acepta esta condición, aparecerá un tercer diálogo en el que puede elegir la cuenta de correo electrónico de Google (Gmail) que quiera. Por defecto aparecerán las cuentas en las que la sesión esté iniciada en su dispositivo, pero también se ofrece la opción de añadir más, tal y como se muestra en la siguiente imagen:

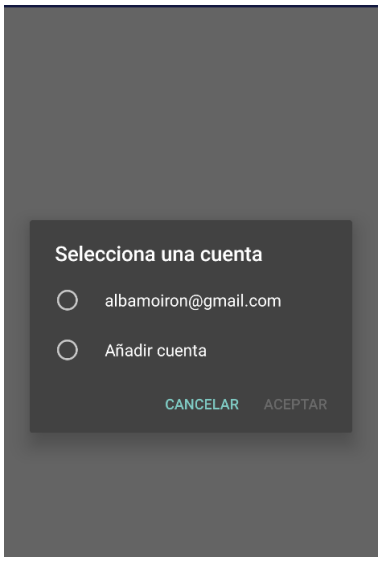

Ilustración 9. Diálogo para seleccionar una cuenta de Google.

Una vez que se han aceptado estos primeros requisitos y se ha creado la hoja, la aplicación guarda la información de credenciales de usuario y los parámetros de la hoja de cálculo, para que cada vez que el usuario acceda a la aplicación no necesite volver a registrarse, sino que siempre se acceda con esta cuenta y se tenga acceso directo a la hoja creada.

Además, se utiliza la cuenta de email seleccionada para mostrarla en el `Navigation Drawer` (barra de navegación lateral) de la aplicación, donde también existe la posibilidad de añadir una foto de perfil. Para ello, es necesario pinchar sobre la imagen que se muestra por defecto, y elegir una entre todas las imágenes que el usuario tenga guardadas en su galería, quedando de la siguiente manera:

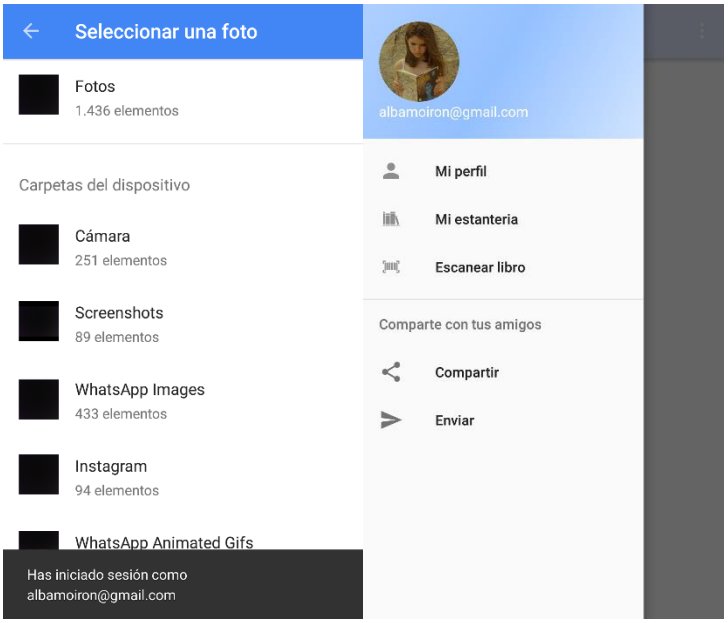

Ilustración 10. Elección de foto de perfil.

Después del registro y la creación de la hoja, se puede acceder a las siguientes pantallas (cuyos accesos se muestran en la imagen anterior):

- **Mi perfil:** Se muestra la información del perfil de usuario, es decir, su foto, el email con el que tiene asociada la aplicación, y la fecha de registro en la misma.

Ilustración 11. Pantalla "Mi perfil".

- **Mi estantería**: Si el usuario todavía no ha añadido ningún libro, aparecerá vacía. Cuando el usuario haya añadido información a su hoja de Google Sheets, en esta pantalla se mostrará una lista con todos los libros que hayan guardado en la misma, mostrando la portada, el título, y un botón en forma de corazón. Si se pulsa sobre este botón, el libro se marcará como "Favorito", y se guardará dicha información en la hoja. Si se pulsa encima de cualquier libro, se mostrará una pantalla con toda la información del mismo, es decir: título, autor, fecha de publicación, puntuación (en forma de ranking de estrellas), género y sinopsis.

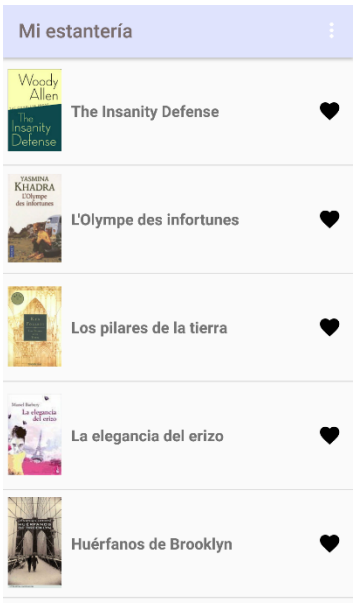

Ilustración 12. Pantalla "Mi Estantería".

- **Escanear libro**: Al pulsar sobre esta opción, primero se comprobará si el usuario tiene la aplicación *Barcode Scanner* instalada en el móvil. Si no es así, se preguntará si desea instalarla, y si el usuario acepta, se le redirigirá al Google Play, el *market* de aplicaciones de Android, donde se la podrá descargar.

  Una vez instalada, si el usuario pulsa sobre esta opción, accederá a una pantalla en la cual dispone de un botón para escanear los códigos de barras de los libros. Cuando se reconoce el libro, mediante su código EAN13, se muestra una pantalla con toda la información del libro, y una pregunta con dos opciones (Sí/No) preguntando al usuario si desea añadir este libro a su hoja de Google Sheets. Si el usuario pulsa sobre la opción "Sí", se guardará toda la información relativa al libro (título, autor, fecha de publicación, etc.…) en la hoja. Si pulsa sobre la opción "No", le llevará de nuevo a la pantalla inicial de la app.

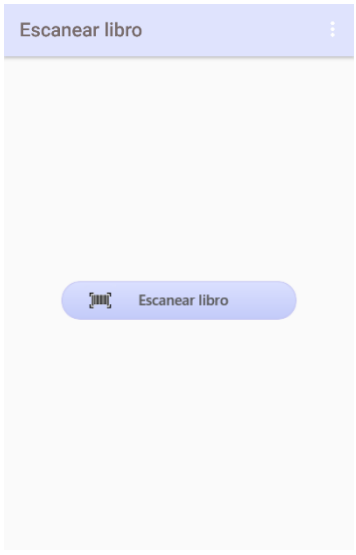

Ilustración 13. Pantalla "Escanear libro".

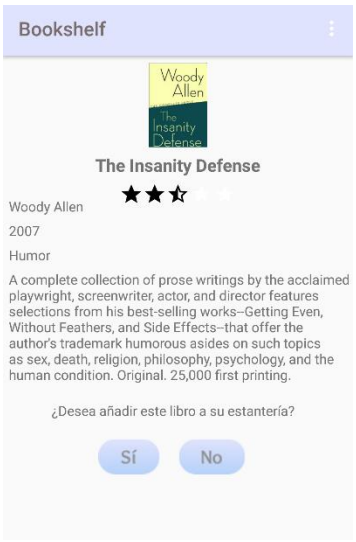

Ilustración 14. Pantalla que muestra información sobre un libro.

- **Compartir:** Pulsando sobre esta opción, aparecerá una lista todas las aplicaciones que permiten compartir contenido, entre las que el usuario tiene instaladas, como pueden ser Facebook, Twitter o Instagram. En el smartphone en el que se han hecho las pruebas, a modo de ejemplo, se muestran las siguientes opciones para compartir:

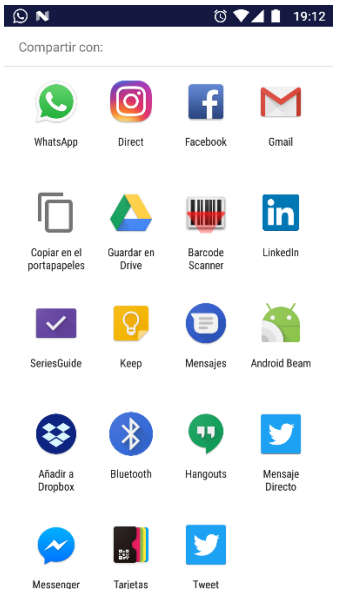

Ilustración 15. Opciones disponibles al pulsar en "Compartir".

- **Enviar:** esta opción accede a las aplicaciones que el usuario tenga instaladas en su móvil, cuyo principal objetivo sea enviar correos electrónicos. En caso de que el usuario solo tenga una app instalada de este tipo, accederá directamente a ella. El asunto del mensaje y el cuerpo del mismo se incluyen directamente en el correo, quedando de la siguiente forma:

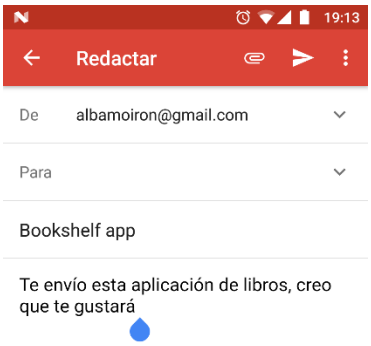

Ilustración 16. Pantalla Gmail que se muestra al pulsar sobre "Enviar".

Todas las pantallas incluyen un Toolbar, es decir, una barra de acciones donde se muestra, en el lado izquierdo, el título de la pantalla en la que se encuentra el usuario, y en el lado derecho un menú ampliado. Pulsando sobre este menú, se podrá acceder a las opciones "Ajustes", "Sobre la aplicación" (cuya pantalla se muestra a continuación) y "Salir de la aplicación". La aplicación también incluye soporte a varios idiomas, en este caso a español, inglés y francés, por lo que, si el usuario cambia el idioma de su dispositivo, también se cambiará el idioma de la aplicación.

Ilustración 17. Pantalla "Sobre la aplicación".

Capítulo 4.

Diseño e implementación de la aplicación.

En este capítulo se profundiza, de una manera más técnica que en el capítulo anterior, en el funcionamiento de la aplicación, destacando el uso de la API de Google Sheets, una de las bases principales de este trabajo. También se hace un resumen de la estructura del código y su contenido más relevante.

4.1. Requisitos funcionales

Los requisitos mínimos, introducidos al inicio de la memoria, que debe cumplir la aplicación, de forma que sea funcional y eficiente son los siguientes:

- La aplicación debe permitir la creación de una nueva hoja de cálculo de Google Sheets cuando el usuario la instala por primera vez en su smartphone. Esta hoja será personal, creándose a través de la cuenta Google que elija el usuario. Una vez se ha creado la hoja, debe quedarse vinculada a la aplicación, sin necesidad de que el usuario vuelva a registrarse. Además, esta hoja, en el momento de crearla, debe aparecer en la carpeta personal de Google Drive del usuario, pudiendo acceder a ella desde la propia app de Google Drive o su página web.

- La aplicación debe permitir escribir en la hoja de cálculo, así como consultar y leer la información que hay en la misma. Esto debe hacerse de forma eficiente, sin que el usuario tenga que esperar mucho tiempo entre cada consulta, o cada vez que quiera actualizar la hoja, aunque esto estará en parte limitado por las condiciones de la conexión de la que se disponga.

- La aplicación debe permitir al usuario escanear el código de barras de un libro, con el fin de obtener información relativa al mismo. Esto se hará utilizando la aplicación *Barcode Scanner.* Si el usuario no la tiene instalada en su dispositivo, se le ofrece la opción de instalarla. En caso de no hacerlo, esta funcionalidad no estará disponible en la app.

- La aplicación debe permitir al usuario acceder a una lista en la que se le muestren todos los libros que tiene guardados en la hoja, permitiéndole marcar aquellos que desee como “Favoritos”.

- La interfaz de la aplicación debe ser intuitiva y sencilla, para que cualquier tipo de usuario pueda entender rápidamente el funcionamiento de la misma.

4.2. Diseño de la arquitectura

La aplicación desarrollada, como se ha comentado anteriormente, sigue una arquitectura de tipo cliente – servidor. El dispositivo móvil, a través de la app, hace peticiones a un servidor web, en este caso el servidor de Google, que se encargará de atender la solicitud del usuario y darle respuesta mediante la API de Google Sheets.

4.2.1. Implementación arquitectura cliente

La aplicación Android desarrollada sigue un esquema que, a grandes rasgos, se compone de clases Java, archivos de recursos (*layouts*) XML y el Manifiesto Android, cada uno de ellos almacenados en directorios específicos del proyecto Android. A continuación se detalla el

funcionamiento básico de cada bloque, qué contiene cada uno y qué uso se le ha dado, para poder entender con lo que se ha trabajado en el desarrollo de la aplicación.

4.2.1.2. Manifiesto

El manifiesto, *AndroidManifest.xml*, es un fichero XML que debe incluirse en toda aplicación Android, y que es la raíz de la misma. Este fichero indica:

- Componentes de la aplicación y relaciones entre ellos.
- Bibliotecas necesarias (además de las propias de Android).
- Permisos que necesita la aplicación para su ejecución.
- Nivel de API mínimo requerido por la app.

Los elementos más destacados del Manifiesto son:

- Nivel 1: elemento `manifest`
  - Atributo `package`: nombre del paquete Java que sirve como base de la aplicación.

- Nivel 2:
  - Elemento `uses-permission`: permisos que necesita la app para funcionar correctamente.
  - Elemento `permission`: permisos de seguridad para limitar el acceso a componentes específicos de la aplicación.
  - Elemento `instrumentation`: permite monitorizar la interacción de la app con el sistema.
  - Elemento `application`: declaración de la aplicación. Dentro de este elemento, se pueden tener los siguientes:
    - Elemento `activity`
    - Elemento `service`
    - Elemento `receiver`
    - Elemento `provider`

    *Estos cuatro elementos se han explicado en el Capítulo 2.

También es en este fichero donde es necesario incluir las claves de API para poder utilizar, en este caso, la API de Google Books. Esto se incluye con la etiqueta `meta-data`, como se introdujo también en el Capítulo 2.

Los elementos que se han incluido en el Manifiesto de esta aplicación son los siguientes:

Tipo de elemento	Elemento	Descripción
manifest	package "com.example.apptfg"	Paquete base de la aplicación
uses-permission	INTERNET	Solicita permiso para poder acceder a Internet
uses-permission	CAMERA	Solicita permiso para poder acceder a la cámara del smartphone
uses-permission	ACCESS_NETWORK_STATE	Solicita permiso para poder acceder al estado de la red
uses-permission	GET_ACCOUNT	Solicita permiso para poder acceder a las cuentas del usuario
application	activity	Etiqueta con la que se añaden todas las pantallas de las que se compone la aplicación
meta-data	com.google.android.gms.version	Par nombre-valor para poder utilizar Google Play Services
meta-data	com.google.android.gms.vision	Par nombre-valor para poder detectar los códigos de barras con el Barcode Scanner
meta-data	com.google.android.books.API_KEY	Par nombre-valor en el que se incluye la API Key para poder utilizar Google Books en la app

Tabla 6. Elementos incluidos en fichero AndroidManifest.xml.

4.2.1.3. Interfaz de usuario

La interfaz de usuario (IU) de una aplicación es todo aquello que el usuario puede ver en su pantalla y con lo que puede interactuar. Android ofrece una variedad de componentes de IU previamente compilados, como objetos de diseño estructurados y controles de IU.

Todos los elementos que componen una interfaz de usuario en Android están desarrollados con objetos `View` y `ViewGroup`. `View` es la clase base de los elementos que forman el IU, y todos los elementos derivan de ella. Un `ViewGroup` es una extensión de la clase `View`, y contiene múltiples hijos `View`, es decir, controles compuestos formados por varios `View` anidados. La IU de cada componente de una aplicación Android se define con una jerarquía de objetos View y `ViewGroup`, tal y como se muestra en la siguiente imagen:

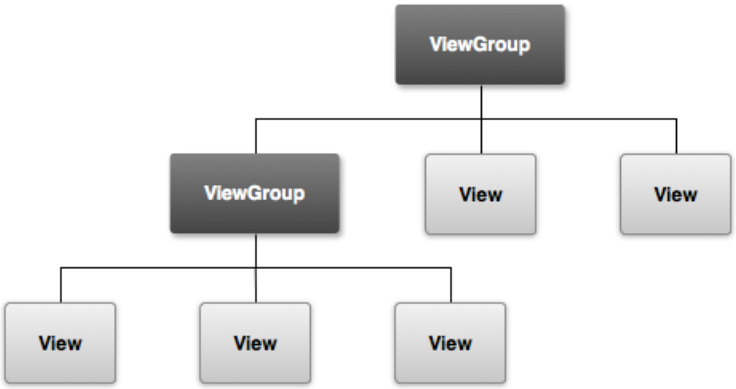

Ilustración 18. Jerarquía de vistas en una aplicación Android. Fuente [12].

La IU de una aplicación Android también se compone de `Activities` y `Fragments`, los cuales se asocian normalmente con una pantalla de la aplicación.

Los `Views` que se han usado en esta app son:

- `TextView`: texto (solo lectura) que puede ser multilínea y al que se le puede aplicar formato.
- `ImageView`: sirve para mostrar imágenes.
- `Button`, `ImaggeButton`: distintos tipos de botones.
- `ListView`: `ViewGroup` que crea y gestiona una lista vertical de `Views` (`TextView`, `ImageView`, etc.).

Cuando se arranca una `Activity`, la pantalla está vacía. Es necesario crear el IU llamando a *setContentView*, que acepta como parámetro un layout o `View`, en el método *onCreate*.

Aunque se pueden implementar en código, se recomienda utilizar *layouts* almacenados en recursos XML. Los layouts son una extensión de la clase `ViewGroup`, que se usa para posicionar

en la pantalla los controles del IU. Los diferentes tipos de layouts que se han usado en la app desarrollada son:

- `FrameLayout`: es el más simple, todas los `View` se sitúan en la esquina superior izquierda.
- `LinearLayout`: se muestran los `Views` alineados de manera horizontal o vertical, según se defina. En este tipo de layout es posible asignar un peso a cada uno de los `View` para controlar el espacio relativo de cada uno.
- `RelativeLayout`: se trata del layout más flexible, ya que permite definir la posición de cada `View` relativa a los demás o a los bordes de la pantalla.

Una vez hecha esta introducción, se pasa a detallar el uso de recursos externos para mejorar la interfaz de usuario de la aplicación.

4.2.1.3.1. Recursos externos

Android soporta la externalización de recursos, es decir, separar del código Java los recursos que no son código como tal. De esta forma, es mucho más sencillo mantener, actualizar y gestionar los mismos. Los recursos se encuentran almacenados en el directorio *res* del proyecto Android, divididos en subdirectorios.

A continuación, se resume brevemente cada uno de los tipos de recursos que se han utilizado en el presente proyecto, y se muestra una tabla resumen de todos los que incluye la aplicación:

- `Drawable`: en este subdirectorio se incluyen las imágenes, preferiblemente en formato PNG, que se usan en el diseño de la aplicación. En caso de tener una misma imagen con varios tamaños, se crean subdirectorios diferenciados para cada uno de ellos.

- `Layout`: permite definir la distribución del IU desacoplado del código. Los principales, como se ha descrito anteriormente, son `LinearLayout`, `RelativeLayout` y `FrameLayout`. Dentro de ellos se pueden utilizar los elementos de clase `View` mencionados arriba (`TextView`, `ImageView`, etc.). Su uso más habitual es definir el IU de una `Activity`, y se "inflan" en el código Java mediante `setContentView`, en el método `onCreate()`.

- `Menu`: permite definir los menús, tanto de actividad como de contexto, en ficheros XML. Posteriormente se "inflan" en el código Java mediante el método `inflate` del servicio `MenuInflator`, normalmente en el método `onCreateOptionsMenu`.

- `Mipmap`: aquí se incluyen los iconos de la aplicación. Al igual que en el caso de `drawable`, se pueden guardar diferentes tamaños de una imagen.

- `Values`: dentro de este subdirectorio se almacenan diferentes ficheros XML, entre los que destacan las cadenas (Strings), colores, dimensiones y arrayas de enteros o de cadenas.

- `Xml`: en este subdirectorio se almacenan recursos arbitrarios que pueden consultarte en tiempo de ejecución como, por ejemplo, los ajustes de la app.

Los diferentes recursos externos que se han creado en esta aplicación, además de las imágenes (guardadas en `Drawable`) y el icono de la aplicación (almacenado en `Mipmap`), son los siguientes:

Tipo de recurso externo	Nombre	Descripción
Layout	activity_about	`LinearLayout` que contiene dos `TextView` y un `ImageView`.
Layout	activity_book_info	`RelativeLayout` que contiene un `LinearLayout` para la `Toolbar`, y varios `TextView` e `ImageView` para mostrar información relativa a un libro.
Layout	activity_create_sheet	`LinearLayout` que contiene un `TextView`
Layout	activity_escanear	`RelativeLayout` que incluye un `Button` para escanear libros y la `Toolbar`
Layout	activity_main	Formado por `Toolbar` y `Navigation Drawer` para la `Activity` principal.
Layout	activity_miperfil	`LinearLayout` que contiene un `CircleImageView` y un `RelativeLayout` donde se incluyen dos `TextView` para la información del usuario.
Layout	activity_spreadsheets	Consta de un `ListView` para poder mostrar el listado de libros que contiene la hoja de Google Sheets.

Layout	fila_lista_miestanteria	Diseño de cada fila de la lista de libros. RelativeLayout con un ImageView, un TextView y un ImageButton.
Layout	main_principal	Incluye Toolbar y resto de IU para la Activity principal.
Layout	nav_header_main	Cabecera del Navigation Drawer, donde se muestra la imagen de perfil y el email de usuario.
Layout	toolbar	Toolbar que se muestra en todas las pantallas de la app
Menu	activity_main_drawer	Opciones menú Navigation Drawer
Menu	menu_main	Menú que se muestra en Toolbar de todas las pantallas
Values	colors	Incluye los valores de color usados en la app
Values	dimens	Incluye los valores de dimensión en pixeles independientes de densidad *dp*
Values	drawables	Incluye los valores de las imágenes drawable
Values	strings	Incluye los valores de strings
Values	styles	Permite que la app mantenga una imagen consistente en las distintas pantallas. Incluye colores y tipos de letra.

Tabla 7. Listado recursos externos empleados en la aplicación.

Algo a destacar sobre lo anterior, es que para poder utilizar un `CircleImageView` [13] en `activity_miperfil`, es necesario añadir una librería de compatibilidad al *build.gradle* del proyecto (no se incluye como opción estándar en Android) de la siguiente forma:

```
compile 'de.hdodenhof:circleimageview:1.2.1'
```

4.2.1.4. Clases Java de la aplicación

Como se ha introducido anteriormente, las aplicaciones Android están escritas bajo el lenguaje de programación Java. Es por ello que las apps se componen de clases Java, las cuales pueden

extender, por ejemplo, `Activities` o `Fragments`, e implementar diversas funciones. Las diferentes clases que componen la presente aplicación son:

About

Extiende la clase `AppCompatActivity`. Esta clase es necesaria para poder incluir una `Toolbar`, y tiene que ser de tipo `Activity` para poder asociarla con una pantalla de IU, en la que, en este caso, se muestra un breve texto con información sobre la aplicación desarrollada y el icono de la misma. Esta clase Java solamente tiene la función de "inflar" su layout asociado, el cual se compone de dos `TextView` y un `ImageView`.

Ajustes

Extiende la clase `PreferenceActivity`. De esta forma, al extender la misma, la `Activity` se encargará de crear la interfaz gráfica de la lista de opciones, según se han definido en su archivo XML asociado. En este caso, se incluye una única opción de "Activar notificaciones".

BookItem

Se trata de una clase que contiene un objeto `BookItem`, el cual representa a un libro, y se compone de la portada y el título del mismo. La clase contiene dos métodos por cada elemento del libro, con el fin de obtener y establecer los valores de título y portada. De esta forma, cuando se quiere rellenar el `ListView` (con el que se muestra el listado de libros almacenados), a través de un adaptador, se puede instanciar esta clase y crear objetos de tipo `BookItem` y utilizar sus métodos.

En resumen, este objeto `BookItem` representa cada una de las filas de la lista de libros que podemos ver en la pantalla *Mi estantería*.

BookAdapter

Extiende la clase `BaseAdapter`. Un adapter (adaptador) es un mecanismo que actúa como puente entre los datos que se obtienen, en este caso, de la hoja de Google Sheets, con el `ListView` donde se muestran todos los libros. Dentro de esta clase, se crea un adaptador al que se le pasa como parámetros el contexto de la aplicación y una lista de String de tipo `BookItem`. Aquí es donde se "infla" el layout que contiene las filas del `ListView`.

Para poder mostrar la portada del libro se ha usado la librería *Picasso* [14], la cual permite mostrar una imagen, guardada en una URL, dentro de un `ImageView`, de forma sencilla y eficiente. Para poder usar esta librería, es necesaria añadirla al *build.gradle* del proyecto de la siguiente forma:

```
compile 'com.squareup.picasso:picasso:2.5.0'
```

GetBookInfo

Esta `Activity` trabaja con la API de Google Books, y tiene el único fin de obtener la información sobre un libro concreto, cuyo código EAN13 recibe a través de la `Activity ScanBook`. Para poder realizar esta acción, se ha incluido un hilo separado del de interfaz de usuario, ya que el acceso a la red puede conllevar un retardo variable e impredecible. Para crear dicho hilo, se ha utilizado la clase `AsyncTask`, que crea una tarea asíncrona con los siguientes manejadores de datos:

- `doInBackground`: aquí se incluye la tarea que se va a ejecutar en un segundo plano, es decir, el método con el que se lee el JSON de la URL proporcionada por la API de Google Books a través del código EAN13 del libro. La URL que devuelve el código y su contenido es similar a lo mostrado en Ilustración 5. Ejemplo URL JSON proporcionada por API Google Books.

- `onPostExecute`: cuando finaliza la ejecución de `doInBackground`, pasa el resultado a este método, en este caso, todos los campos relevantes del libro: título, autor, fecha de publicación, categoría, sinopsis, puntuación y portada. Después de asignar los valores a todos estos datos, hace una llamada a la clase `BookInfo`, para que muestre toda esta información al usuario en una pantalla independiente.

BookInfo

Extiende `AppCompatActivity`, y se encarga de recoger los resultados de un libro que el usuario ha buscado o escaneado (cuyos datos le llegan a través de la `Activity` anterior), y muestra toda su información por pantalla. También incluye dos botones para que el usuario elija si quiere añadir el libro a su hoja de Google Sheets o no.

Si el usuario decide que quiere guardar dicho libro en su hoja personal, se hará una llamada a la clase `Spreadsheets`, la encargada de leer y escribir en la hoja de Google Sheets del usuario. En caso de pulsar sobre la opción “No”, se llevará al usuario de nuevo a la pantalla principal.

MainActivity

Se trata de la pantalla principal de la aplicación. Será la primera con la que el usuario se encuentre al abrir la misma y a la cual volverá cada vez que pulse el botón “Atrás” desde cualquiera de las otras pantallas que componen la app.

Esta `Activity` incluye la creación del `Navigation Drawer`, es decir, el panel lateral de navegación de la app, desde el cual se puede acceder a la mayoría de pantallas (Mi Perfil, Mi

estantería, Escanear Libro). Visualmente, solo incluye una breve frase que se mostrará siempre que se acceda a esta pantalla.

Cuando se instala la aplicación y se accede a la misma, esta clase es la encargada de comprobar si es la primera vez que el usuario la usa, y si es así, lanza la opción para que el usuario pueda crear una hoja de Google Sheets, es decir, hace una llamada a la clase `CreateSpreadsheets`. Una vez creada la hoja, esta clase guarda en las Preferencias (un mecanismo para salvar datos de la aplicación, `SharedPreferences`) que la app ya ha sido iniciada y que se ha creado la hoja, para no volver a hacerle esta pregunta al usuario en un futuro.

Esta `Activity` también se encarga de escribir en el `Navigation Drawer` la cuenta de Gmail que el usuario ha elegido, y gestiona el cambio de foto de perfil en la cabecera del mismo. Cuando el usuario pulsa sobre la foto que viene por defecto, podrá acceder a la galería de su dispositivo, y elegir cualquier imagen como foto de perfil, pudiendo volver a modificarla en cualquier momento que desee. Si el usuario elige una foto de perfil, esta se pasa mediante un `Intent` a la clase `Mi Perfil`, para que pueda mostrarla en su IU.

Mi Perfil

`AppCompatActivity` cuya función es mostrar al usuario los datos sobre su perfil. Una parte importante de la pantalla se rellena con la foto de perfil que el usuario ha elegido, y debajo de esta se muestra la cuenta de email que tiene asociada con la aplicación y la fecha en la que se ha realizado el registro en la misma.

ScanBook

Esta `Activity` es la encargada de escanear los códigos de barras de los libros. Para poder acceder a la información sobre un libro, el formato del código de barras debe ser EAN13, es decir, un código EAN (European Article Number) formado por 13 dígitos, donde los tres primeros indican el código del país, los cuatro (o cinco) siguientes se relacionan el código de la empresa, y los siguientes, hasta completar los 12, se corresponden con el código de producto. El último número es el dígito de control.

Primero, es necesario destacar que para poder implementar esta funcionalidad se ha añadido la librería de compatibilidad *ZXing* [15] al *build.gradle* del proyecto de la siguiente forma:

```
compile 'me.dm7.barcodescanner:zxing:1.9'
```

La primera vez que se accede a esta pantalla, se comprobará si el usuario dispone de la aplicación *Barcode Scanner*. En caso contrario, se ofrecerá la posibilidad de acceder a Google Play para su descarga. Una vez que el usuario haya obtenido esta app, podrá escanear códigos de barras de

libros cada vez que acceda a la `Activity`. Por lo tanto, si el usuario no acepta la descarga de dicha aplicación, no podrá acceder a la funcionalidad de escanear códigos de barras de libros.
Si el código escaneado no está en formato EAN13, o ha ocurrido algún error escaneando, se avisará al usuario y se le dará la opción de intentarlo de nuevo. Si el escaneo se resuelve con éxito, se pasará el código EAN, mediante un `Intent`, a la `Activity BookInfo`, donde, como se ha comentado, se muestra toda la información relativa al libro escaneado.

CreateSpreadsheets

Es la `Activity` encargada de crear la hoja personal del usuario, asociada a su cuenta Google. A esta `Activity` solo se accede en una ocasión: cuando el usuario descarga la aplicación y la arranca por primera vez.

Se encarga de crear las credenciales de usuario a través de la cuenta que este elija, y llama a un servicio de Google para crear la hoja. Por defecto, la hoja se crea con una serie de cabeceras en la primera fila, de forma que el usuario pueda identificar, cuando almacene información, qué celda pertenece a cada tipo de dato, es decir, título, autor, fecha publicación, etc. También guarda, a través de las preferencias, el identificador de la hoja (`sheetId`) y las credenciales de usuario, de forma que cada vez que se acceda a la aplicación, se recuperen las mismas y el usuario no tenga que volver a registrarse.

Spreadsheets

Es la `Activity` encargada de interactuar con la hoja creada previamente, es decir, de acceder a su información en modo lectura y de escribir en la misma cada vez que el usuario desee añadir un nuevo libro. Además, es la encargada de extender el `ListView` (donde se muestra el listado de libros disponibles en la hoja) y relacionarlo con el adaptador de la clase `BookAdapter` para mostrar la información relativa a los libros.

AñadirFavorito

Extiende de la clase `Activity`, pero no tiene una IU asociada, solamente realiza una acción. En este caso se lleva a cabo el mismo procedimiento que la clase `Spreadsheets`, pero solamente para actualizar la hoja. Cuando se marca un libro como “Favorito”, pulsando sobre la imagen del corazón en la lista de libros disponibles en la hoja personal, esta clase accede a la hoja de Google Sheets del usuario y actualiza el campo de “Favorito” poniendo un “Sí” en él.

En el siguiente apartado se profundizará más sobre el funcionamiento de estas tres últimas clases Java que trabajan con la API de Google Sheets.

Adicionalmente, al pulsar sobre las opciones “Compartir” y “Enviar por email” del `Navigation Drawer`, aparecerá una lista de aplicaciones instaladas en el dispositivo del usuario, en función

de cuál de las dos opciones se pulse, como se ha mostrado en Ilustración 15. Opciones disponibles al pulsar en "Compartir". e Ilustración 16. Pantalla Gmail que se muestra al pulsar sobre "Enviar". Estas dos opciones se basan en `Intents`; no se relacionan con `Activities`.

4.2.2. Implementación arquitectura servidor con API Google Sheets

En este punto se profundizará en la arquitectura del servidor, es decir, el funcionamiento de Google Sheets, su uso y su aplicación en este proyecto. Dentro de este apartado también entraría el uso de la API de Google Books, pero por simplicidad y porque ya se ha introducido su uso en apartados anteriores, este se centrará en la API de Google Sheets.

Antes de explicar el funcionamiento de cada una de las clases Java que trabajan con la API de Google Sheets, es necesario hacer una introducción sobre qué se necesita para poder usar la misma en el código:

1) Acceso a la Consola Google Developers. El primer paso es acceder a esta URL y crear un nuevo proyecto, indicando el nombre de la app y otros datos relativos al desarrollador.

2) Una vez finalizado este primer paso, es necesario activar la API de Google Sheets, desde la siguiente pantalla:

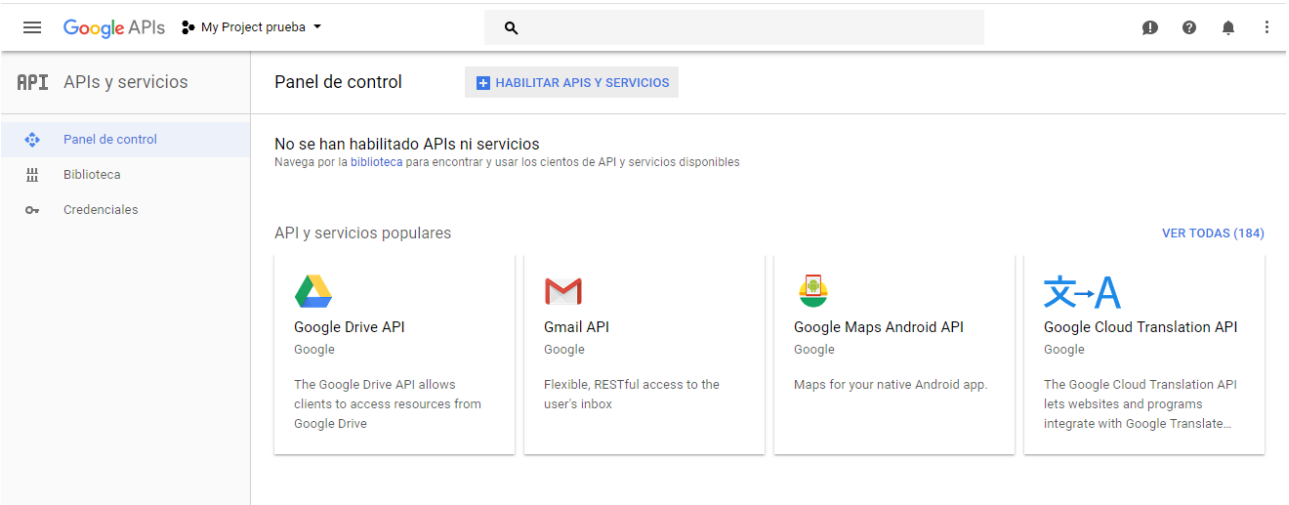

Ilustración 19. Pantalla principal Consola Google Developers.

Pulsando sobre "Habilitar APIs y Servicios", se mostrarán todas las APIs que proporciona Google, teniendo que pulsar sobre "Habilitar" en la opción de "Google Sheets API".

3) Cuando se pulsa sobre "Habilitar", aparecerá una nueva pantalla en la que se informa de que es posible que se necesiten crear credenciales para su uso, como se muestra a continuación:

Ilustración 20. API Google Sheets habilitada en Consola Google Developers.

4) Si se pulsa sobre "Crear credenciales", aparecerá una nueva pantalla en la que hay que elegir el tipo de aplicación desde donde se usará la API, en este caso Android, y a qué tipo de datos de usuario se accederá, en este caso, a datos pertenecientes a un usuario de Google. Después de marcar estas opciones, se pasará a la creación de un usuario de OAuth2.

Haciendo un breve paréntesis, OAuth2 es un protocolo de autorización que permite a terceros, es decir, a clientes, acceder a contenidos propiedad de un usuario. En esta aplicación, esto se relaciona con la necesidad de acceder a la cuenta Google de un usuario para poder trabajar con la API. Un escenario OAuth2 tiene tres partes claramente diferenciadas:

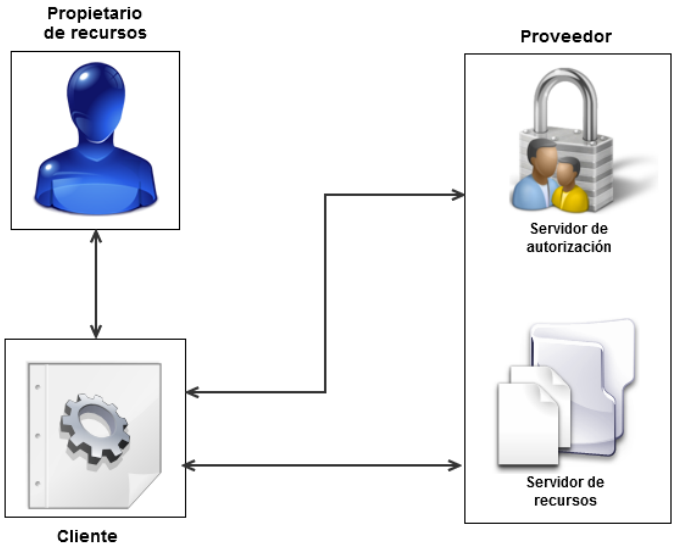

Ilustración 21. Escenario OAuth2. Fuente [16].

Para crear las credenciales OAuth2 se necesita especificar el nombre del paquete del proyecto (`package`), el cual se puede encontrar en el `AndroidManifest`, como se ha explicado anteriormente, y la huella digital de certificado de firma SHA-1 (Secure

Hash Algorithm). SHA-1 es un algoritmo seguro con el que se firma la aplicación Android, y se utiliza para poder hacer uso de las API de Google bajo OAuth2.

Para obtener la huella digital de certificado de firma, es necesario acceder desde el software Android Studio, al apartado *Gradle projects*, pulsar sobre el proyecto (en este caso *MyApplication2 (root)*), y a continuación acceder a *Tasks/Android* y hacer doble click sobre *signingReport* para que se ejecute. Una vez haya finalizado la ejecución, se mostrará la clave SHA-1, tal y como se puede ver en la siguiente imagen:

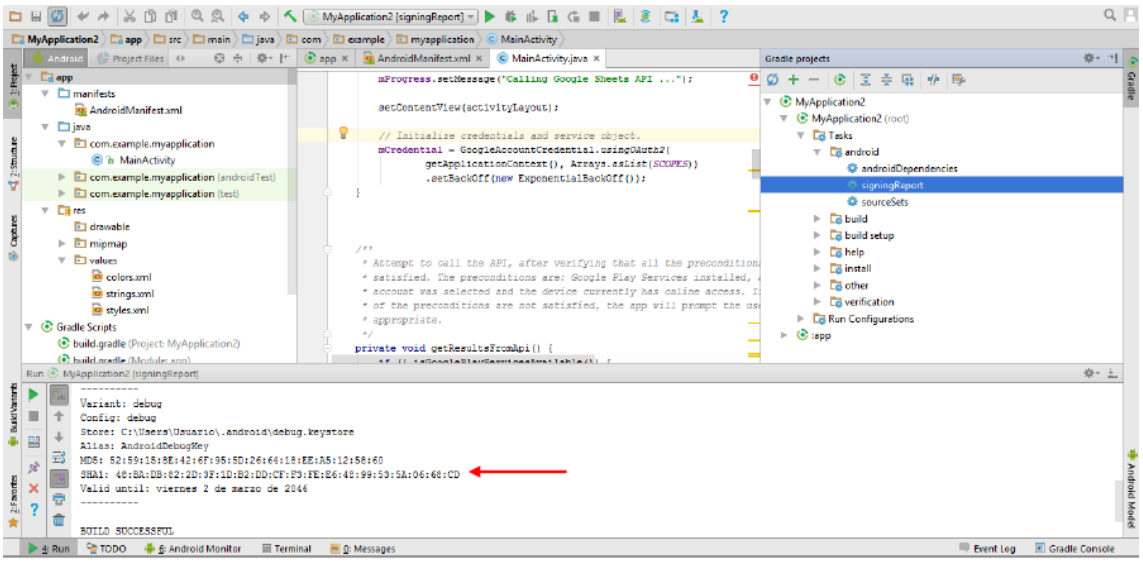

Ilustración 22. Obtención huella digital de certificado de firma SHA-1.

Esta huella se incluirá en la información solicitada en la Consola de Google Developers. Una vez hecho esto, se pulsará sobre "Crear ID de cliente". De esta forma se tendrá acceso a la API de Google Sheets desde el código.

5) Por último, dentro de Android Studio, es necesario acceder al fichero *build.gradle*, y añadir las librerías necesarias para el uso de la API, tal y como se muestra a continuación:

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:25.0.1'
    compile 'com.google.android.gms:play-services-auth:10.2.0'
    compile 'pub.devrel:easypermissions:0.2.1'
    compile('com.google.api-client:google-api-client-android:1.22.0') {
        exclude group: 'org.apache.httpcomponents'
    }
    compile('com.google.apis:google-api-services-sheets:v4-rev461-1.22.0') {
        exclude group: 'org.apache.httpcomponents'
    }

}
```

Ilustración 23. Modificación build.gradle para añadir librerías necesarias.

Con estos primeros pasos, ya se tiene todo lo necesario para empezar a trabajar con la API de Google Sheets. A partir de aquí, se explicarán los pasos llevados a cabo en cada una de las clases Java que trabajan con esta API:

1) Creación de una hoja de Google Sheets: clase `CreateSpreadsheets`

Cuando la aplicación se descarga en el dispositivo y se usa por primera vez, se pedirá al usuario permiso para crear una nueva hoja de cálculo de Google Sheets vinculada a su cuenta Google. Por lo tanto, la misión de esta clase, `CreateSpreadsheets`, es crear una hoja a partir de las credenciales de usuario y almacenar la información relativa a la misma, con el fin de que en próximos accesos a la app el usuario no tenga que registrarse de nuevo. De esta forma, la hoja se queda vinculada a la aplicación y se carga de forma automática cuando se vuelva a entrar en la misma.

En primer lugar, se hará un resumen de las variables necesarias para trabajar con la API de Google Sheets en esta primera interacción con la misma:

Variable	Valor inicialización
`GoogleAccountCredential mCredential`	`GoogleAccountCredential.usingOAuth2( getApplicationContext(), Arrays.asList(SCOPES)) .setBackOff(new ExponentialBackOff());`
`com.google.api.services.sheets.v4.Sheets mService`	`null`
`String PREF_ACCOUNT_NAME`	`accountName`
`String[] SCOPES`	`{SheetsScopes.SPREADSHEETS}`
`String spreadsheet_id`	-
`String spreadsheet_url`	-
`int REQUEST_ACCOUNT_PICKER`	`1000`
`int REQUEST_AUTHORIZATION`	`1001`
`int REQUEST_GOOGLE_PLAY_SERVICES`	`1002`
`int REQUEST_PERMISSION_GET_ACCOUNTS`	`1003`

Tabla 8. Inicialización variables relevantes clase CreateSpreadsheets.

En referencia a la tabla anterior, la variable `mCredential` sirve para administrar y seleccionar la cuenta de Google del usuario, y se usa como parámetro del método `MakeRequestTask`, el

cual extiende un hilo en segundo plano `AsyncTask`. Por otra parte, la variable `mService` se usa para definir un servicio de Google Sheets (v4), y se inicializa en principio a *null*. Una vez se hayan hecho todas las comprobaciones previas a la creación de la hoja, las cuales se comentarán a continuación, se entrará en un hilo de tipo `AsyncTask`, donde se inicializará esta variable de la siguiente forma:

```
private class MakeRequestTask extends AsyncTask<Void, Void, List<String>> {
    private Exception mLastError = null;

    MakeRequestTask(GoogleAccountCredential credential) {
        HttpTransport transport = AndroidHttp.newCompatibleTransport();
        JsonFactory jsonFactory = JacksonFactory.getDefaultInstance();
        mService = new com.google.api.services.sheets.v4.Sheets.Builder(
                transport, jsonFactory, credential)
                .setApplicationName("Google Sheets")
                .build();
    }
```

Ilustración 24. Inicialización servicio dentro del hilo AsyncTask.

Otra de las variables más relevantes es `SCOPES`, la cual define el alcance del uso de la API. Inicializándola con el valor `{SheetsScopes.SPREADSHEETS}`, se está dando a la aplicación permiso de escritura y acceso a las propiedades de la hoja de cálculo. En caso de querer que solo se permitiese la función de lectura, se modificaría de la siguiente forma: `{SheetsScopes.SPREADSHEETS_READONLY}`.

Las variables `spreadsheet_id` y `spreadsheet_url` se usan para almacenar el ID identificativo de la hoja de cálculo que se va a crear, y para almacenar la URL completa de dicha hoja, respectivamente. En cuanto a las últimas cuatro variables que se muestran en la tabla, sirven para solicitar acceso a la selección de cuentas de usuario y servicios de Google Play.

A continuación, se muestra un diagrama de cómo se trabaja con los primeros métodos utilizados en la clase antes de la creación de la hoja de Google Sheets:

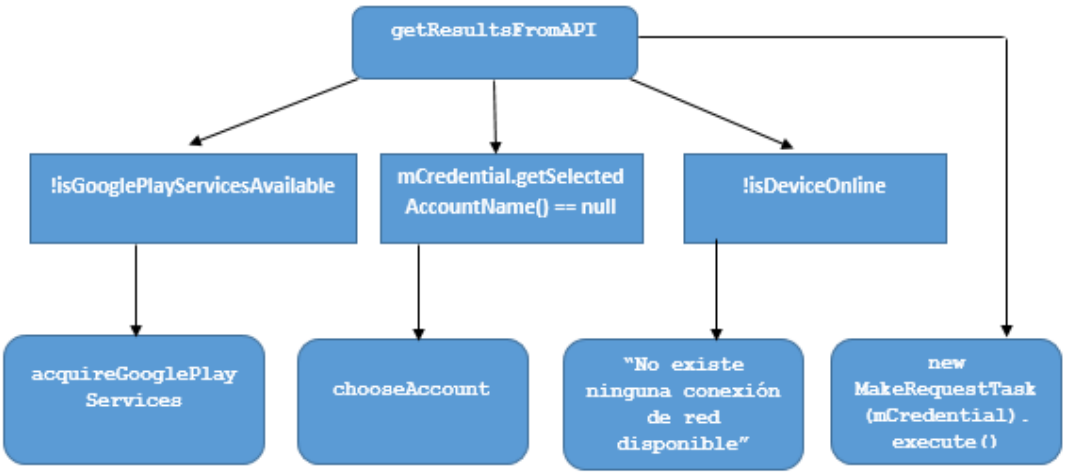

Ilustración 25. Diagrama métodos requisitos previos a creación hoja de cálculo.

Profundizando en el funcionamiento de cada método:

- `getResultsFromAPI`: es el primer método al que se accede en esta clase, después del `onCreate` de la `Activity`. Tiene cuatro condiciones, tal y como se muestra en el diagrama, las cuales se explican a continuación:

    - Si los servicios de Google Play no están disponibles, (!`isGooglePlayServicesAvailable`) se accede al método `acquireGooglePlayServices`, el cual intenta resolver la falta, caducidad o invalidez de Google Play Services a través de un diálogo que se muestra al usuario, con el fin de dar aviso del problema e indicarle que es necesaria su instalación para poder usar la app. Por lo tanto, si este requisito no se cumple, no se podrá crear la hoja.

    - Si la vinculación de la cuenta con las credenciales de la API es nula, se llama al método `chooseAccount`, el cual intenta realizar dicha vinculación. Si el nombre de la cuenta se hubiese declarado de manera específica previamente, se utilizaría esa misma. Al no ser así, se mostrará un diálogo para que el usuario pueda seleccionar la cuenta que desea entre las que tiene vinculadas a su dispositivo. Una vez el usuario haya elegido la cuenta, se guardarán sus datos a través de preferencias, de modo que la app se quede vinculada con la cuenta de usuario seleccionada.

    - Si el dispositivo desde el que se está usando la aplicación no tiene conexión, se avisará al usuario mediante un texto en pantalla. Por lo tanto, si no se dispone de ningún tipo de conexión, no se podrá continuar con la creación de la hoja de cálculo.

    - Por último, si se cumplen todos los requisitos anteriores, se creará una tarea en segundo plano, es decir, un hilo asíncrono, pasándole por parámetro la credencial de la API que se ha declarado al inicio. Es en este paso donde se inicializará la variable `mService` como se ha mostrado en la *Ilustración 24. Inicialización servicio dentro del hilo AsyncTask.*

- Si en el método anterior, `getResultsFromAPI`, se ha hecho la llamada a la tarea en segundo plano y se ha inicializado la credencial de la API, se pasará a crear la hoja de cálculo dentro de esta tarea, ya que su retardo puede ser variable e impredecible, por lo que es preferible hacer la llamada desde dicho hilo.

  La hoja se crea dentro del método `CreateSpreadsheets`, y es aquí donde se utilizarán las colecciones y términos introducidos en API Google Sheets. Además de crear la hoja, en este método también se realiza una primera escritura en la misma, para

establecer los títulos de cada una de las columnas que contendrán información sobre un libro.

Para la creación de la hoja, se siguen los pasos que se muestran a continuación:

```
com.google.api.services.sheets.v4.model.Spreadsheet mSpreadsheet, newSpreadSheet;
mSpreadsheet = new Spreadsheet();
SpreadsheetProperties spreadsheetProperties = new SpreadsheetProperties();
spreadsheetProperties.setTitle("Mi estantería");
mSpreadsheet = mSpreadsheet.setProperties(spreadsheetProperties);

newSpreadSheet = mService.spreadsheets()
        .create(mSpreadsheet)
        .execute();

spreadsheet_id = newSpreadSheet.getSpreadsheetId();
spreadsheet_url = newSpreadSheet.getSpreadsheetUrl();
```

Ilustración 26. Código para creación de una nueva hoja de cálculo.

Donde es necesario cargar el siguiente paquete:

```
com.google.api.services.sheets.v4.model.Spreadsheet
```

De esta forma, se podrá crear la hoja, así como acceder a las propiedades de la misma y darle un título. Después de darle nombre, se solicita la creación de una nueva hoja bautizada con el mismo, a través del servicio inicializado anteriormente, y la colección `spreadsheets`.

A continuación, se obtienen el ID y la URL de la hoja de cálculo, y posteriormente, esta información se almacena a través de las preferencias para que la aplicación se quede vinculada a esta cuenta de usuario y a la hoja que se ha creado.

Como se ha comentado en la introducción sobre la API, en API Google Sheets, es necesario establecer un rango en notación A1, tanto para lectura como para escritura. En este caso se ha establecido en base al número de columnas que se necesitan para insertar la información del libro, es decir, una columna por cada parámetro del volumen (título, autor, fecha, …), además de un campo para marcar con 'Sí/No' si el libro se ha añadido como favorito y otro para indicar el estado del libro (leído/pendiente/leyendo). El rango establecido es "Sheet1!A1:I", el cual hace referencia a las 9 primeras celdas en la primera fila de la hoja "Sheet1".

También es necesario establecer un array de arrays, `values`, donde el array exterior representa todos los datos, y cada array interior representa la dimensión principal. Cada elemento del array interior se corresponde con una celda de la hoja. Por esto mismo, otro parámetro importante es `majorDimension`, la dimensión principal de los valores, que en este caso se ha elegido el valor 'ROWS' (filas).

Para escribir en la primera fila de la hoja los nombres identificativos de cada una de las columnas pertenecientes a la misma, se ha utilizado la colección `spreadsheets.values`, y, concretamente, el método `append`.

Una vez finalizado el método, se llama a `onPostExecute`, el cual muestra al usuario un mensaje informándole de que se ha creado una nueva hoja de cálculo personal, que servirá como base de datos de la aplicación.

Esta clase ya no se volverá a utilizar más en todo el procedimiento. Solo se volvería a entrar en ella en caso de que el usuario desinstalase la aplicación de su dispositivo y volviese a instalarla de nuevo.

2) Lectura de una hoja de Google Sheets: clase `Spreadsheets`

Esta clase tiene dos misiones principales: leer y escribir información en la hoja de cálculo, y mostrar un `ListView` en el que se añadirán, después de la lectura, todos los libros disponibles en la hoja.

Primero, es necesario declarar de nuevo las variables `mCredential`, `mService` y `SCOPES`, de la misma forma que se hacía en la clase `CreateSpreadsheets`. También se crea un `String` por cada parámetro del libro, es decir, autor, título, fecha, etc., cuya información se recoge a través del `Intent` que se envía desde la clase `BookInfo` (cuando el usuario pulsa sobre el botón de añadir el libro a su hoja).

Para poder trabajar con el `ListView`, es necesario declarar una variable de tipo `List<BookItem>` para almacenar los datos de la hoja, una variable de tipo `BookItem`, que representa el título y la portada del libro, y otra variable de tipo `BookAdapter`, necesaria para poder vincular los datos que se obtienen con el `ListView`, como se ha comentado anteriormente.

De la misma manera que en la clase `CreateSpreadsheets`, es necesario crear una tarea en segundo plano para realizar las tareas de lectura y escritura. A continuación se explican ambos métodos, en los cuales se ha declarado el mismo rango en notación A1. El rango establecido es "Sheet1!A2:I", el cual hace referencia a nueve columnas, empezando por la segunda fila de la hoja, puesto que la primera está ocupada por las cabeceras establecidas en el método `CreateSpreadsheets`. Estas cabeceras no son de interés en la lectura de los datos, ni se desea modificarlas, por lo que se dejan fuera del rango.

- `getDataFromApi`: Este método tiene la misión de leer los datos que se tienen en la hoja. Concretamente, se recogen los valores en la posición 0, el título del

libro, y en la posición 6, la portada del mismo, pues son los datos que se desean mostrar en el `ListView`.

Para la lectura, además del rango, es necesario crear una variable para guardar los resultados que se van a leer y a devolver al final del método, de tipo `List<String>`, y una variable de tipo `ValueRange`, que como se ha comentado en API Google Sheets, hace referencia a los datos dentro del rango establecido.

Para leer los datos se hace uso de la colección `spreadsheets.values`, y en concreto, del método `get`, pasándole por parámetro el ID de la hoja y el rango establecido. También se tiene, como se ha mencionado en casos anteriores, un array de arrays, `values`, donde el array exterior representa todos los datos, y cada elemento del array interior se corresponde con una celda de la hoja. Dentro del bucle se van añadiendo los resultados deseados a la variable `results`, que se devuelve al final del método y será el parámetro del método `onPostExecute`, donde se asignarán estos resultados a la variable de tipo `List<BookItem>` que se ha declarado al inicio, y se relacionaran con el `Adapter` para poder rellenar la lista.

```
private List<String> getDataFromApi() throws IOException {

    String range = "Sheet1!A2:I";
    List<String> results = new ArrayList<String>();
    ValueRange response = mService.spreadsheets().values()
            .get(sheet_id, range)
            .execute();
    List<List<Object>> values = response.getValues();

    if (values != null) {
        for (List row : values) {
            results.add(row.get(0) + "," + row.get(6));
        }
    }
    return results;
}
```

Ilustración 27. Método lectura getDataFromApi.

- `setDataToApi`: Este método se encarga de escribir la información sobre un libro en la hoja personal del usuario. De la misma forma que el método anterior, se declaran las variables `range` y `values`, además de una variable de tipo `List<Object>`, inicializada como un `ArrayList`, donde se irán añadiendo los valores a escribir en la hoja. Aquí se hace uso de las variables de tipo String declaradas al inicio de la clase, que representan los parámetros del libro. Para escribir en la hoja, se hace uso de la colección `spreadsheets.values`, y en concreto, del método `append`, pasándole por parámetros el ID de la hoja, el rango establecido y un objeto de tipo `ValueRange`. Además, se utiliza el

método `setValueInputOption`, que como se ha introducido en API Google Sheets, no analiza la entrada, simplemente la inserta como un `String`.

```
values.add(data1);
ValueRange valueRange = new ValueRange();
valueRange.setMajorDimension("ROWS");
valueRange.setRange(range);
valueRange.setValues(values);

ValueRange body = new ValueRange().setValues(values);

AppendValuesResponse response =
        mService.spreadsheets().values().append(sheet_id, range, body)
                .setValueInputOption("RAW")
                .execute();
```

Ilustración 28. Extracto método escritura setDataToApi.

3) Actualización de una hoja de Google Sheets: clase `AñadirFavorito`

Esta clase también interactúa con la hoja de Google Sheets, en este caso, para cambiar el estado del campo "Favorito" de la hoja (inicialmente vacío) una vez que el usuario pulsa sobre la imagen del corazón en el `ListView`. Si el usuario pulsa sobre dicha imagen, se hará una llamada a esta clase y se pondrá un "Sí" en el campo "Favorito". Para ello, se repite lo mismo que se ha hecho en el método de lectura `setDataToApi`, es decir, se usan las mismas variables y la colección `spreadsheets.values`, pero en este caso el método utilizado es `update`. Este método, después de buscar en la hoja el título que coincida con el que el usuario ha seleccionado añadir a favoritos, actualiza el campo dándole el valor de favorito. Se adjuntan las líneas de código que realiza este método:

```
ValueRange valueRange = new ValueRange();
valueRange.setMajorDimension("ROWS");
valueRange.setRange(range);
valueRange.setValues(values2);

ValueRange body = new ValueRange().setValues(values2);

UpdateValuesResponse result =
        mService.spreadsheets().values().update(sheet_id, range2, body)
                .setValueInputOption("RAW")
                .execute();
```

Ilustración 29. Actualización campo Favorito hoja Google Sheets.

Por lo tanto, a partir de lo visto en este capítulo, se puede afirmar que se ha conseguido implementar los requisitos detallados en el apartado 4.1.Requisitos funcionales.

Capítulo 5.

Pruebas y validación.

Una vez finalizado el desarrollo software, se ha pasado a realizar pruebas sobre el mismo, para comprobar que la aplicación no se cierra de forma inesperada, y que todo funciona según lo esperado y se cumplen los requisitos funcionales mínimos.

5.1. Pruebas de la aplicación

Realizar pruebas sobre el funcionamiento de la aplicación es de vital importancia, ya que puede que no se cumplan de forma correcta los requisitos mínimos establecidos para el desarrollo de la misma, o que, cuando se produce un error en la app, el gestor de aplicaciones de Android la cierre (mostrando un mensaje de *log* donde se indica que la app se ha cerrado de forma inesperada).

A continuación se detallan las validaciones más importantes que se han hecho y, en caso de haber encontrado algún tipo de error, la solución empleada:

Requisito	Tipo de prueba	Validación
Creación de la hoja de Google Sheets en la clase `CreateSpreadsheets`	Se comprueba, desde el Google Drive del usuario, que efectivamente se ha creado la hoja	Correcta
Escritura de datos en la hoja desde la clase `CreateSpreadsheets`	Se comprueba, a través de Google Drive, que la hoja contiene la información que se ha añadido	Correcta
Lectura de datos en la hoja desde la clase `CreateSpreadsheets`	Se comprueba, mediante *logs*, que se están leyendo los datos deseados	Correcta
Escritura y lectura desde la clase `Spreadsheets`	Se comprueba que las acciones de leer y escribir se realizan de forma correcta	Incorrecta
Almacenar credenciales usuario e ID hoja cálculo	Se comprueba que las credenciales de usuario y el ID de la hoja se guardan de forma correcta	Incorrecta
Pulsar botón "Atrás"	Se comprueba que pulsando sobre el botón "Atrás", se redirige a la `Activity` principal	Incorrecta
Búsqueda de libros introduciendo título por teclado	Se comprueba que esta funcionalidad se realiza de forma correcta	Incorrecta

Tabla 9. Principales pruebas y validaciones app.

**Todas las validaciones marcadas como incorrectas en la tabla finalmente se solucionaron.*

En cuanto a las validaciones incorrectas, es decir, aquellas en las que se han encontrado fallos, se han solucionado de la siguiente forma:

Los métodos de lectura y escritura, en una primera versión de la app, se encontraban dentro de la clase `CreateSpreadsheets`, por simplicidad y para comprobar que todo funcionaba correctamente. Cuando se pasaron estos métodos a una clase independiente, `Spreadsheets`, se comprobó que dichas acciones no se llegaban a completar nunca, puesto que no se establecía conexión con la hoja del usuario mediante la API. Esto se debía a que se estaba pasando a dichos métodos las variables `mCredential` y `mService` creadas en `CreateSpreadsheets`. Esto se ha solucionado creando de nuevo estas variables dentro de esta clase, e inicializándolas de la misma forma que se hacía en la clase que crea la hoja. Esto va relacionado con el siguiente error que se encontró, es decir, que las credenciales de usuario y el ID de la hoja no se estaban almacenando correctamente. Para ello, se hizo uso del objeto `SharedPreferences`, guardando mediante el mismo la cuenta del usuario y el ID de la hoja. De esta forma, en la clase `Spreadsheets` se recuperan estos parámetros, necesarios para la lectura y escritura. Combinando ambas soluciones, se ha conseguido realizar con éxito las acciones deseadas.

Otro error que se encontró es que cuando se pulsaba el botón "Atrás" en `Spreadsheets`, la aplicación se cerraba de forma inesperada. Para ello, se ha añadido el método `onBackPressed`, y dentro de él un `Intent` para llamar a la `Activity` principal, `MainActivity`. Una vez que se comprobó que esto funcionaba correctamente, se añadió el mismo método a las demás clases que tenían una IU asociada, de forma que este error no volviese a aparecer jamás.

En cuanto a la búsqueda de libros, la cual ahora mismo se hace mediante el escaneo de los códigos de barras de un libro, también se tenía inicialmente una funcionalidad en el `Toolbar` del `MainActivity` relacionada con esto. Se trataba de un icono de una lupa, que al pulsar sobre la misma, se podía introducir por teclado el título del libro a buscar. El problema que surgió está relacionado con la API de Google Books que, como se ha comentado anteriormente, se encuentra en una versión experimental, por lo que tiene muchas limitaciones y no ofrece demasiada flexibilidad. Por tanto, cuando se introducía un título por teclado, la API devolvía todos los resultados que contenían dicho título. La carga computacional que suponía esto era insostenible, ya que la tarea tardaba bastante tiempo en realizarse, lo que se traduce en que el usuario tendría que esperar un largo tiempo hasta obtener los resultados para, después de ello, tuviese que buscar el que realmente le interesaba, entre una lista de títulos muy numerosa (en varias pruebas se llegaron a encontrar hasta 200 resultados diferentes). Es por ello que se decidió eliminar esta funcionalidad e incluirla como una posible línea futura de trabajo.

Capítulo 6.

Conclusiones y líneas futuras.

En este capítulo se muestran las conclusiones del proyecto, una vez se ha finalizado el mismo. También se analizan las potenciales líneas de trabajo futuras del software, con el fin de introducir mejoras sobre el mismo.

Tras finalizar el proyecto es interesante incluir las conclusiones y reflexiones sobre el mismo, y así analizar los objetivos y el conocimiento obtenido.

Los objetivos principales que se habían establecido al inicio de este trabajo, mencionados en el apartado 1.2. Objetivos del Capítulo 1 han sido completados satisfactoriamente:

- La aplicación crea de manera correcta una hoja de cálculo de Google Sheets asociada a la cuenta de Google elegida por el usuario, y se puede acceder a la misma a través del Google Drive de su dueño.

- La aplicación permite escribir en la hoja creada, así como leer la información que hay en la misma.

- La aplicación permite al usuario escanear el código de barras de un libro y obtener la información relativa al mismo mediante su código EAN13.

- La interfaz de la aplicación es sencilla e intuitiva.

Además de esto, también se ha conseguido otra serie de funcionalidades:

- Las acciones de crear la hoja, escribir y leer, se realizan de forma rápida y eficiente, sin necesidad de que el usuario tenga que esperar mucho tiempo a que se completen.

- El usuario puede marcar como favoritos los libros que desee. Esto se traduce en que se ha conseguido implementar la función de actualizar una celda de la hoja de cálculo, además de las de lectura y escritura.

- El usuario puede compartir, a través de otras apps que tenga instaladas en su móvil, contenido sobre la app BookShelf.

Por lo tanto, se ha conseguido desarrollar una plataforma de acceso a una base de datos personal almacenada en la nube, que no ocupa espacio en el teléfono móvil, gratuita en un principio, y a la cual se puede acceder en cualquier momento en tiempo real, tanto desde el dispositivo como desde la propia aplicación de Google Drive o su página web.

Podría decirse que sería una buena alternativa al uso de servidores propios, de forma que pequeños empresarios pudiesen utilizarla en sus comercios o empresas como base de datos, sin necesidad de adquirir o alquilar un servidor, ni de contratar a una persona especializada en la materia para supervisar su correcto funcionamiento y mantenimiento.

6.1. Líneas futuras

Aunque los objetivos principales que se habían impuesto al comienzo del proyecto se han completado con éxito, es importante buscar nuevas metas y mejoras para la aplicación desarrollada. A continuación se analizan las posibles líneas de trabajo futuras que se podrían aplicar a este proyecto:

- Añadir la funcionalidad de buscar libros por palabras clave. Para ello, se utilizaría una API diferente a la de Google Books, ya que, como se ha comentado en el capítulo anterior, se encuentra en una versión experimental, por lo que está muy limitada y no permite demasiada flexibilidad.

- Mejora de la interfaz. Dividir el listado de libros en listados independientes clasificados como “Leyendo”, “Pendientes de lectura” y “Leídos”, de forma que el usuario pueda acceder a la información de una forma más organizada. También estaría bien incluir una nueva sección de “Recomendaciones”, de forma que, en función del tipo de libros que el usuario añada a su hoja, se le muestren recomendaciones de estilo o género similar. Se podría incluir, en el apartado “Mi perfil”, el número de libros que el usuario ha leído, así como el número de libros pendientes, a modo de resumen.

- Mejora de la funcionalidad de compartir. Se podría añadir la funcionalidad de compartir la información de un libro concreto, no solo sobre la app en sí como se ha hecho en este caso.

- Crear una versión de la app para el sistema operativo iOs, ya que las APIs de Google incluyen documentación para el mismo y de esta forma se ampliaría el mercado de la aplicación.

- Publicar la aplicación desarrollada en el *market* de Android, es decir, en Google Play.

Capítulo 7.

Entorno socioeconómico, presupuesto y marco regulador.

En este capítulo final se hace un análisis del impacto socioeconómico esperado de la aplicación desarrollada, y del marco regulador que se tendría que cumplir en caso de una futura comercialización de la misma. Se incluye también un análisis detallado del presupuesto estimado del proyecto.

7.1. Entorno socioeconómico

El avance de Internet y el uso de dispositivos como smartphones o tablets está cambiando el mundo en el que vivimos. Se ha estimado que en 2017 la cifra de usuarios de smartphones en el mundo ha alcanzado los 2,3 billones, número que se espera que siga aumentando [17]. En España, en el año 2016, los smartphones representaban un 87% de los teléfonos móviles del país [18].

En este contexto, en España están surgiendo un gran número de empresas que se dedican al desarrollo de aplicaciones móviles, ya que es un negocio que tiene una gran demanda y se espera que cada vez crezca más. En el año 2015, el número de usuarios de apps móviles era de 27,7 millones [18] y, en 2016, el uso de apps en nuestro país ocupaba un 89% del tiempo dedicado a los smartphones. Analizando todos estos datos es lógico pensar que desarrollar aplicaciones móviles hoy en día supone una gran oportunidad de negocio.

A su vez, muchos pequeños negocios no han logrado adaptarse a este gran cambio, ya sea por falta de información o por falta de dinero. Como se ha comentado al inicio de esta memoria, y en relación con este proyecto, adquirir un servidor propio y mantenerlo, supone un gran coste para un negocio pequeño como puede ser una librería familiar, una tienda de ropa o una ferretería de barrio. Es importante que estos negocios también puedan transformarse y no quedarse atrás en cuanto a innovación y tecnología en un mundo globalizado y digitalizado.

En cuanto a esto, se puede destacar la iniciativa del Ministerio de Agenda Digital de España [19][2], el cual ha aprobado recientemente dos programas para impulsar la transformación digital de las PYMES (Pequeñas y Medianas Empresas), donde también se incluyen a las microempresas, es decir, aquellas con un máximo de 10 empleados, donde se pueden encuadrar las empresas objetivo de este proyecto. Tanto en España como en Europa en general, se está dando mucha importancia a esto, y ambos tienen la digitalización de los negocios como un objetivo dentro de su agenda digital para un futuro próximo.

El objetivo que tiene el proyecto que se ha desarrollado es precisamente ese, brindar a pequeños empresarios la oportunidad de digitalizar en cierto modo su negocio, y que, de esa manera, el método para registrar su stock o sus cuentas sea mucho más sencillo y puedan realizarlo a través de un smartphone, pues analizando las cifras mencionadas anteriormente, es altamente probable que dispongan de uno. Con esto se podrían ahorrar mucho trabajo y tiempo en la actualización y mantenimiento de su stock o cuentas, además de poder lograrlo por un precio muy bajo.

También es destacable que el tipo de digitalización que se propone puede resultar sencillo e intuitivo para los empresarios, ya que, como se ha mencionado al inicio del documento, el uso

[2] http://www.minetad.gob.es/es-ES/GabinetePrensa/NotasPrensa/2017/Paginas/ElMinisteriodeAgendaDigitalapruebadosprogramasparaimpulsarlatransformaci%C3%B3ndigitaldelaspymes.aspx
*Se incluye aquí debido a que, por su larga extensión, no se podía incluir la URL completa en la bibliografía

de Google es algo muy extendido en la sociedad, y para usar el tipo de app que se propone no es necesario tener ningún tipo de conocimiento técnico más allá de conocer el uso de un smartphone y de los servicios que ofrece Google.

7.2. Presupuesto

En este apartado se hará un análisis detallado del presupuesto estimado del proyecto y los recursos empleados.

7.2.1. Recursos empleados

- **Recursos humanos:**
  - 1 ingeniero senior
  - 1 ingeniero junior

- **Recursos materiales**:
  - 1 ordenador portátil de gama media con SO Windows, que debe cumplir los siguientes requisitos [20] para poder instalar Android Studio:
    - Windows 7/8/10 (32 o 64 bits).
    - 2 GB de RAM (8 GB de RAM recomendado).
    - 2 GB de espacio libre mínimo (4 GB recomendado).
    - Resolución mínima de 1.280 x 800.
    - Java 8.
    - 64 bits y procesador Intel (emulador).
  - 1 smartphone de gama media con versión Android 7.0.

- **Licencias de software:**
  - Microsoft Office 2017
  - Windows 10 Home
  - Android Studio v3.1
  - Google Sheets
  - Barcode Scanner

7.2.2. Presupuesto detallado

1.- Autor: Alba Moirón Alén

2.- Departamento*:* Ingeniería Telemática

3.- Descripción del Proyecto:

- Titulo: Diseño y desarrollo de una app basada en Android y Google Spreadsheets

- Duración (meses): 8 meses

- Tasa de costes indirectos: 20%

4.- Presupuesto total del Proyecto: 9.599,68 €

5.- Desglose presupuestario: A continuación se recoge el presupuesto estimado para la realización del proyecto. Se desglosa en coste del personal, coste del hardware y coste del software y, finalmente, se muestra el resumen del presupuesto añadiéndole los costes indirectos, es decir, el IVA.

7.2.2.1. Coste personal

En primer lugar, se analizan los costes asociados a la mano de obra. Se incluye primero una tabla con el desglose de las actividades realizadas por el ingeniero junior y después de esta, otra tabla con los costes asociados a las dos personas involucradas en el proyecto.

La dedicación del ingeniero senior no se muestra en el desglose por horas, pero es necesario destacar que ha dedicado, de forma aproximada, unas 25 horas a la labor de supervisión del trabajo.

Tareas	Dedicación ingeniero junior (horas*)
Planteamiento del proyecto y descarga entorno desarrollo	14
Documentación y pruebas sobre aplicación ejemplo	20
Programación de la aplicación	182
Diseño final de la interfaz	88
Pruebas y control de errores	30
Conclusiones y estudio de futuras líneas de trabajo	8
Redacción de la memoria final	122

Tabla 10. Desglose del tiempo empleado por el personal en cada tarea.

Tipo de mano de obra	Cantidad (horas)	Coste unitario (€/hora)	Coste total (€)
Ingeniero senior	25	35	875
Ingeniero junior	464	18	8.352
		Total	**9.227**

Tabla 11. Coste personal.

7.2.2.2. Coste hardware

A continuación se muestran los costes asociados a la adquisición del material hardware necesario para realizar el proyecto. El coste imputable de cada dispositivo se ha calculado siguiendo la siguiente fórmula:

$$Coste\ imputable\ (€) = \frac{Meses\ uso}{Periodo\ depreciación} * Coste\ dispositivo$$

Descripción dispositivo	Cantidad	Coste (€)	Uso (meses)	Periodo de depreciación (meses)	Coste imputable (€)
Ordenador portátil ASUS	1	450	8	72	50
Smartphone BQ Aquaris X	1	260	8	24	86,7
				Total	**136,7**

Tabla 12. Coste hardware.

Como se puede comprobar, el coste total imputable, que es igual a 136,7 €, es resultado de la suma de los costes individuales.

7.2.2.3. Coste software

De igual forma que en el apartado anterior, se procede a mostrar el desglose de los costes asociados al software utilizado, donde cabe destacar que todas las licencias mencionadas anteriormente son gratis, excepto las que se muestran en la siguiente tabla:

Descripción	Coste (€)
Microsoft Office Professional 2016	188,99
Windows 10 Home	46,99
TOTAL	**235,98**

Tabla 13. Coste software.

7.2.2.4. Presupuesto final

Por último, se muestra el coste total del proyecto, resultado de la suma de los costes descritos anteriormente, y sumándole el Impuesto sobre el Valor Añadido (IVA):

CONCEPTO	COSTE TOTAL (€)
Personal	9.227
Hardware	136,7
Software	235,98
Costes indirectos (20%)	1.919,94
Total (sin IVA)	**9.599,68**
Total (con IVA)	**11.519,62**

Tabla 14. Presupuesto final.

7.3. Marco regulador

Por último, se analiza el marco regulador aplicable sobre la implementación de la aplicación desarrollada en este Trabajo Fin de Grado.

7.3.1. Ley Orgánica 15/1999, de Protección de Datos de Carácter Personal

La Ley de Protección de Datos de Carácter Personal tiene como objetivo garantizar y proteger la intimidad y demás derechos fundamentales de las personas físicas frente al riesgo que para ellas supone un posible uso indiscriminado de sus datos personales.

El deber de información a las personas físicas de las cuales se vaya a obtener cualquier tipo de datos personales, previo al tratamiento de los mismos, es uno de los principios fundamentales sobre los que se asienta la *Ley Orgánica de Protección de Datos de Carácter Personal (LOPD)*, haciendo referencia a ello en el artículo 5.

Debido a la extensión de esta ley [21], se citan los artículos que más se relacionan con este proyecto:

- ***Artículo 5. Derecho de información en la recogida de datos**: "[1] Los interesados a los que se soliciten datos personales deberán ser previamente informados de modo expreso, preciso e inequívoco:*

  a) De la existencia de un fichero o tratamiento de datos de carácter personal, de la finalidad de la recogida de éstos y de los destinatarios de la información.

  b) Del carácter obligatorio o facultativo de su respuesta a las preguntas que les sean planteadas.

  c) De las consecuencias de la obtención de los datos o de la negativa suministrarlos.

  d) De la posibilidad de ejercitar los derechos de acceso, rectificación, cancelación y oposición.

  e) De la identidad y dirección del responsable del tratamiento o, en su caso, de su representante."

- ***Artículo 10. Deber de secreto**: "El responsable del fichero y quienes intervengan en cualquier fase del tratamiento de los datos de carácter personal están obligados al secreto profesional respecto de los mismos y al deber de guardarlos, obligaciones que subsistirán aun después de finalizar sus relaciones con el titular del fichero o, en su caso, con el responsable del mismo."*

- ***Artículo 12. Acceso a los datos por cuenta de terceros:** "[2] La realización de tratamientos por cuenta de terceros deberá estar regulada en un contrato que deberá constar por escrito o en alguna otra forma que permita acreditar su celebración y contenido, estableciéndose expresamente que el encargado del tratamiento únicamente tratará los datos conforme a las instrucciones del responsable del tratamiento, que no los aplicará o utilizará con fin distinto al que figure en dicho contrato, ni los comunicará, ni siquiera para su conservación, a otras personas."*

- ***Artículo 26. Notificación e inscripción registral**: "[1] Toda persona o entidad que proceda a la creación de ficheros de datos de carácter personal lo notificará previamente a la Agencia de Protección de Datos."*

Teniendo en cuenta los artículos mencionados, para que la aplicación desarrollada cumpla los mismos, deben adoptarse las siguientes medidas:

- **Comunicación al usuario del acceso a su cuenta Google**: en la app desarrollada se pide permiso al usuario para acceder a su cuenta de Google. Sería necesario incluir una pantalla adicional con las condiciones de uso, donde el usuario pueda leer, de forma más detallada, el uso que se le darán a sus datos y pueda aceptar si está de acuerdo con las mismas.

- **Compromiso de no proporcionar los datos a terceros**: esto se podría añadir en la pantalla mencionada anteriormente, donde se comunica al usuario las condiciones de uso de la app.

- **Deber de secreto de los datos recopilados**: en esta aplicación este punto se cumplirá de forma estricta, tal y como se mencionará en las condiciones de uso.

- **Entrega del fichero a la Agencia de Protección de Datos**: antes de la creación de ficheros de datos personales, sería necesario notificar a la Agencia de Protección de Datos.

Además de informar del acceso a la cuenta de Google y del tipo de datos que recogerá la aplicación en sí, también sería necesario informar de la Política de Privacidad propia de Google [22], la cual se explicará brevemente en el siguiente apartado.

7.3.2. Política de Privacidad de Google

La Política de Privacidad de Google describe el tipo de datos, a partir de una cuenta de usuario de Google, que recoge la compañía y con qué fines llevan a cabo la recogida de los mismos, además de indicar cómo utilizan dichos datos y las opciones que ofrecen en cuanto a seguridad.

Google recolecta información sobre los servicios que el usuario utiliza y de qué forma usa los mismos, por lo que a continuación se citan algunos puntos incluidos en el apartado de su Política de Privacidad sobre la recogida de este tipo de información:

- **Información del dispositivo**. *"Recopilamos información específica del dispositivo (como tu modelo de hardware, la versión de tu sistema operativo, identificadores de dispositivo únicos e información de la red móvil, incluido el número de teléfono). Google podrá asociar los identificadores de tu dispositivo o tu número de teléfono a tu cuenta de Google."*

- **Datos de registro**. *"Cada vez que utilizas nuestros servicios o consultas nuestro contenido, obtenemos y almacenamos determinada información en los registros del servidor de forma automática. Estos datos incluyen: información detallada sobre cómo utilizas nuestro servicio (por ejemplo, tus consultas de búsqueda), datos telefónicos como, por ejemplo, tu número de teléfono, […], la dirección IP, información relativa a tu dispositivo como, por ejemplo, fallos, actividad del sistema, ajustes del hardware, tipo de navegador, idioma del navegador, fecha y hora de tu solicitud y URL de referencia, cookies, que permitirán identificar tu navegador o tu cuenta de Google."*

- **Datos sobre tu ubicación física**. *"Cuando utilizas los servicios de Google, podemos recopilar y procesar información sobre tu ubicación real. Empleamos diferentes tecnologías para determinar la ubicación, como la identificación de la dirección IP, el*

> *sistema GPS y el uso de otros sensores que pueden proporcionar a Google, por ejemplo, información sobre dispositivos cercanos, puntos de acceso Wi-Fi y antenas de telefonía móvil."*

Por esto, además de que el usuario acepte las condiciones de la aplicación en sí, realmente la misma va ligada a la Política de Privacidad de Google, por lo que habría que informar sobre ella, aunque se supone que la ha aceptado en su día, en el momento que decidió crear una cuenta de Google. Aun así, como se comentaba en la *Ley de Protección de Datos*, es necesario informar al usuario sobre toda la información que se recogerá sobre sus datos, tanto en la aplicación como cuando se conecta al servicio Google Sheets, ya que puede que no esté de acuerdo con ceder ese tipo de información.

Es necesario destacar que las cláusulas de privacidad de una cuenta gratuita de Google y de una cuenta de empresa son diferentes. Lo mencionado anteriormente tiene un carácter general, pero, en el caso de las empresas que pagan por este servicio, se ofrece un nivel de seguridad mayor. En caso de contratar el paquete Gsuite [2], Google asegura un nivel de seguridad empresarial máxima. Ofrece confidencialidad directa total, encriptando el contenido y el correo electrónico, e incluye sistemas de autenticación más seguros que en el caso de cuentas gratuitas, por lo que sus datos están mucho más protegidos, dando incluso la opción de activar la prevención de pérdida de datos para Gmail y Google Drive.

BIBLIOGRAFÍA

[1] C. Smith, «Expanded Ramblings,» 22 Noviembre 2017. [En línea]. Available: https://expandedramblings.com/index.php/gmail-statistics/. [Último acceso: 01 Febrero 2017].

[2] Google, «Gsuite Google,» [En línea]. Available: https://gsuite.google.es/intl/es/pricing.html?tab_activeEl=tabset-companies. [Último acceso: 03 Febrero 2017].

[3] Google, «Developers Google,» 2 Agosto 2017. [En línea]. Available: https://developers.google.com/sheets/api/quickstart/java. [Último acceso: 28 Enero 2017].

[4] Google, «Android Developers,» [En línea]. Available: https://developer.android.com/guide/platform/index.html?hl=es-419. [Último acceso: 28 Enero 2017].

[5] IDC, «IDC,» Mayo 2017. [En línea]. Available: https://www.idc.com/promo/smartphone-market-share/os. [Último acceso: 20 Diciembre 2017].

[6] D. García, «andro4all,» 13 Noviembre 2017. [En línea]. Available: https://andro4all.com/2017/11/distribucion-android-noviembre-2017. [Último acceso: 20 Diciembre 2017].

[7] Google, «Developers Google,» [En línea]. Available: https://developers.google.com/sheets/api/. [Último acceso: 25 Enero 2017].

[8] Google, «Developers Google,» [En línea]. Available: https://developers.google.com/books/?hl=es-419. [Último acceso: 4 Febrero 2018].

[9] ISSN, «ISSN (International Standard Serial Number),» [En línea]. Available: http://www.issn.org/es/comprender-el-issn/usos-del-issn/la-identificacion-con-el-codigo-de-barras-ean-13/. [Último acceso: 10 Febrero 2018].

[10] «Goodreads,» [En línea]. Available: https://www.goodreads.com/. [Último acceso: 10 Enero 2018].

[11] «iTunes,» [En línea]. Available: https://itunes.apple.com/es/app/ireaditnow/id349773058?mt=8. [Último acceso: 10 Enero 2018].

[12] Google, «Developers Android,» [En línea]. Available: https://developer.android.com/guide/topics/ui/overview.html?hl=es-419. [Último acceso: 4 Febrero 2017].

[13] hdodenhof, «Github,» [En línea]. Available: https://github.com/hdodenhof/CircleImageView. [Último acceso: 1 Febrero 2018].

[14] «Github,» [En línea]. Available: http://square.github.io/picasso/. [Último acceso: 12 Enero 2018].

[15] «Github,» [En línea]. Available: https://github.com/zxing/zxing. [Último acceso: 31 Enero 2018].

[16] A. González, «The Game of Code,» 31 Julio 2012. [En línea]. Available: http://www.thegameofcode.com/2012/07/conceptos-basicos-de-oauth2.html. [Último acceso: 4 Febrero 2017].

[17] Statista, «Statista,» Junio 2016. [En línea]. Available: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/. [Último acceso: 02 Febrero 2018].

[18] ditrendia, «amic,» 2016. [En línea]. Available: http://www.amic.media/media/files/file_352_1050.pdf. [Último acceso: 11 Febrero 2018].

[19] T. y. A. D. Misterio de Energía, «minetad,» [En línea]. Available: http://www.minetad.gob.es/es-ES/GabinetePrensa/NotasPrensa/2017/Paginas/index.aspx. [Último acceso: 20 Enero 2018].

[20] A. Ardións, «AndroidStudioFaqs,» 5 Mayo 2016. [En línea]. Available: https://androidstudiofaqs.com/conceptos/android-studio-requisitos-minimos. [Último acceso: 12 Febrero 2018].

[21] A. E. -. B. d. Estado, «www.boe.es,» 14 Diciembre 1999. [En línea]. Available: http://www.boe.es/buscar/doc.php?id=BOE-A-1999-23750. [Último acceso: 25 Enero 2018].

[22] Google, «Google,» 18 Diciembre 2017. [En línea]. Available: https://www.google.es/intl/es/policies/privacy/. [Último acceso: 15 Febrero 2018].

[23] D. Stuttard y M. Pinto, The Web Application Hacker's Handbook 2nd ed., Indianapolis: John Wiley & Sons, Inc., 2011.

[24] [En línea]. Available: http://www.minetad.gob.es/es-ES/GabinetePrensa/NotasPrensa/2017/Paginas/index.aspx.

[25] R. Meier, Professional Android Application Development, Indianapolis: John Wiley & Sons, 2009.

[26] A. C. García, Introducción a la Administación de Empresas, Madrid: Editorial Civitas, 2008.

ANEXO: EXTENDED ABSTRACT

I. Introduction

Very commonly, small businesses such as libraries, clothes or stationary shop, feel the need to consult or modify their stock information in real time, but do not own any structure network or server for that matter. In this case, the solution is the use of more rudimentary methods like the update of the information by hand, which implies an added difficulty and time to the job, as well as the risk of making mistakes.

However, the fact of acquiring and maintaining a specific server has a very high monetary cost, which many of these small businesses are not able or willing to afford. Given their sales volume, the spent could not be profitable. Besides the server's price, retailers might face the need to hire an additional employee with specific skills for the management of the server, with its additional cost.

Nowadays, access to Google is so extended among population that, in the mid-2017, more than 1.2 billion [1] people owned an email account (Gmail) of the company. Having a Google account implies the possibility to access all the services it offers. These include Google's office software, which has online apps among which Google Sheets, a spreadsheets application that can be related to Windows' Excel, could be remarked.

This project intends to offer a simple and intuitive alternative to the use of the servers, either personal or specific, and give the possibility to collect the stock information of a small business in a Google Sheets spreadsheet. This way, the owners and/or employees of the company can access, modify and consult this data in real time. Hence, every time they make a sale or receive new products, they will be able to reflect it in this database that will be stored in Google's servers.

During this project, a book management app has been developed as an example of the use that Google Sheets' technology could be given. It is a client-server app, where the client is Android and the server is based in Google Sheets and the given Google's APIs. The person who uses this app can search books and store their information (title, author, publication date, etc.) on his personal spreadsheet, as well as consulting it when necessary. This could be done both from the smartphone and the computer, as the spreadsheet stays uploaded on the associated Google Drive account given by the user.

The main goals of this project are:

- The app must allow the creation of a new Google Sheets spreadsheet when the user installs it for the first time on his smartphone.
- The app must allow writing on the spreadsheet, as well as consulting the information on it.

- The app must allow the user to scan a book's barcode, to obtain all its information.
- The app's interface must be simple and intuitive, so that any user can quickly understand it.

II. State of Art

II.I. Client-server technology

Nowadays' services are mainly made of two parts, client and server. Most of the times, the first offers a mobile version, since there is a big number of users that usually access services only through their smartphones. This client version is normally developed for the main operating systems, Android and iOs. For this project, the server is based on Google Sheets, Google Books and their respective APIs.

i. Client: Android

Android is a Linux based operating system, mainly oriented to mobile devices, like smartphones and tablets, although its use on smartwatches, TVs and even automobiles is growing these days. It is a free, open source OS. To develop Android apps, only Java knowledge is needed, together with the download of the programming environment Android Studio, which is available for the main operating systems (Windows, Ubuntu, macOS). The development on Android is component oriented, therefore an app will be made of one or several of these. The main components of an Android app are:

- **Activity:** normally representing a screen with user interface. It is implemented as a subclass of the `Activity` class. Every Android app must have at least one component of this type.
- **Services:** component that models an activity that is executed in the background, normally because of its computing workload or the need of executing this task while performing another one. This component is not associated to a graphic user interface.
- **Broadcast Receivers:** used for receiving and answering broadcast messages. It does not perform any task, it only responds to a notification sent by some activity that processes the information. Most of the messages are created by the system, such as the 'low battery' notifications. The applications can also generate their own messages and broadcast them to notify other apps.
- **Content Providers:** through these content providers, other apps can consult or modify, if allowed, the data. This information can be stored in a database, in the system, or in any other kind of storage. In summary, their function is to provide a set of data of an app that will make them available for another one.

Every Android app is the result of a combination of one or more of these components. They all must be explicitly declared in the manifest (*AndroidManifest.xml* file), which function is to inform the system about such components, as well as declaring the minimum Android version that can run the app (known as the minimum API level), identifying the required user permissions, or including the needed hardware attributes (like camera, location or Bluetooth, for instance).

ii. Server: Google Sheets and Google Books APIs

i. Google Sheets

Google Sheets, available since June 2006, is a free and online spreadsheets application that allows the users to create, modify and give format to these sheets. It also allows several people to work on them simultaneously. It can be accessed from a computer, smartphone or tablet, and even offers the possibility to consult the spreadsheet with no network connection needed. It includes a chat where several users can interact while modifying the document in real time. The changes are automatically saved while being made, and the history of modifications with older versions and recent updated can be consulted. Moreover, the application allows to open, edit and save files in Excel format.

Its use is possible through Google Sheets API, which allows the code to create and modify spreadsheets. To work with it, only a Google account is required, and the acceptance of the required permissions regarding access to contacts and storage.

i.i. Google Sheets API

Google Sheets' API [7], which is currently on its fourth version, allows the reading and modification of any aspect of a spreadsheet. As an introduction to the API, the following are the common terms that a user can find on it:

- **Spreadsheet ID:** All the methods included in the API require the parameter `spreadsheetId`, used for the identification of the spreadsheet that is being used. This ID, which is made of numbers, letters and some special characters, is in spreadsheet's URL, placed between */d* and */edit,* as shown in bold down below:

  `https://docs.google.com/spreadsheets/d/`**`1qpyC0XzvTcKT6EISywvqESX3A0MwQoFDE8p-Bll4hps`**`/edit#gid=0`

- **Page ID:** Inside a spreadsheet, each of its sheets have a different title (that must be unique) and an ID. `sheetId` is used for specifying in which sheet the user's in. It is also located in the spreadsheet's URL as the value of the parameter `gid`. The URL's structure and the `sheetId` location is shown next:

  `https://docs.google.com/spreadsheets/d/`*`spreadsheetId`*`/edit#gid=`***`sheetId`***

- **A1 Notation:** Some API methods require the ranges in A1 notation. This is a `String` like `Sheet1!A1:B2`, which refers to a group of cells in the spreadsheet and is generally used in formulas.

- **`ValueRange` object:** Data within a range of the spreadsheet. It can have several fields:

    - `range`: the range the values cover. It must be in A1 notation.
    - `majorDimension`: major dimension of the values.
    - `values[]`: the data to be read or written.

The methods that the object `ValueRange` can offer are the following: `append`, `batchClearbatchClearByDataFilter`, `batchGetbatchGetByDataFilter`, `batchUpdate`, `batchUpdateByDataFilter`, `clear`, `get`, `update`.

The API offers two ways of interacting with the spreadsheet:

- **Reading or writing cell values only**. This can be done through the collection `spreadsheets.values`, that offers the following methods:

Range access	Reading	Writing
Single range	`spreadsheets.values.get`	`spreadsheets.values.update`
Multiple ranges	`spreadsheets.values.batchGet`	`spreadsheets.values.batchUpdate`
Appending		`spreadsheets.values.append`

    - **Reading**

To read data from a spreadsheet, it is necessary to have the spreadsheet's ID (`spreadsheetId`) and the A1 notation of the ranges. There are three optional parameters to control the output format: `majorDimension`, `valueRenderOption` and `dateTimeRenderOption` (see Table 15).

- **Read a single range:** using `spreadsheets.values.get` request, which returns a `ValueRange` object.

- **Read multiples ranges:** for reading multiples discontinuous ranges, use `spreadsheets.values.batchGet` request, that allows to specify any number of ranges to retrieve. This returns an `BatchGetValueResponse` object, that contains the spreadsheet's ID and a list of `ValueRange` objects.

- **Writing**

To write to a spreadsheet, it is needed the spreadsheet's ID (`spreadsheetId`) and the A1 notation of the ranges. The data to be written in an appropriate request body object is also required. Updates require a valid `ValueInputOption` parameter, which control if the incoming `String` is parsed or not, as described in Table 16.

The methods to write one or multiples ranges of the sheet are:

- **Writing a single range**: using `spreadsheets.values.update` request.
- **Writing multiples ranges:** in order to read multiple discontinuous ranges, its needed to use a `spreadsheets.values.batchGet` request, where it is necessary to specify the `spreadsheetId`, a `valueInputOption` and one or more `ValueRange`.

- **Reading or writing any aspect of the spreadsheet,** through the `spreadsheets` collection. In addition to the data of its cells, a sheet includes cell format, cell margins, ranges with name, ranges with protection and additional formats. These are some of the types of data that control the appearance and functioning of a spreadsheet. Trough the method `batchUpdate` any of these elements can be updated.

i.ii. Google Books API

The Google Books API [8], which is on its first experimental version, offers the possibility to access information about the books, consult the availability of eBooks or even access to full texts. For using it from a client's app, it is required to acquire an API key, and write it on the app's `Manifest`. To get an API key, it is necessary to access Google's Developers Console[3]. It is only needed to look for the API and activate it, associating it to the app's project, and then write this API key into the `Manifest`, in the following way:

```
<meta-data
        android:name="com.google.android.books.API_KEY"
        android:value = "AIzaSyCpYxz5556X4UzPV6rFxxxxsCs_Q_x"/>
```

To retrieve the information about a book, JSON (*JavaScript Object Notation*) has been used. This is a format for data exchange. First, the user must scan the book's barcode. Through its EAN13 code [9], Google Books API will find the book and retrieve its information. In this case, the code searches the book's title, author, publishing date, rating, abstract and category.

[3] https://console.developers.google.com/apis/

III. Application description

When the app (which has been given the name ***Bookshelf***) is installed and launched, a dialog appears where the user is asked if the creation of a new Google Sheets spreadsheet is wanted. This is a necessary requirement for the use of the app. If the user presses "No", a message will be shown informing that the creation of the sheet is necessary, and the option will be shown again. Once the user has pressed the option "Yes" on the dialog, a request for accessing the user's contacts appears. If the user accepts this condition, a third dialog will show, where the user can choose a Gmail account.

Once this first requirements have been accepted and the spreadsheet has been created, the app will save the user's credentials and the spreadsheet's parameters. This way, every time the user access the app, he will not have to register or login again. Moreover, the chosen email account will be used to be shown in the `Navigation Drawer` (the lateral navigation bar) of the app, where there is also a possibility of adding a profile picture.

After the registration and the creation of the spreadsheet, the following screens can be accessed:

- **My profile:** it shows the user's profile, meaning the profile picture, the email address associated to the app and the registration date.

- **My Bookshelf:** if the user still has not added any book, it will be empty. When the user adds data to the spreadsheet, this screen will show a list of all the books that have been saved on the sheet, showing their cover, title and a heart-shaped button. If this button is pressed, the book will be marked as "Favorite", being this information also saved on the sheet. If the user presses on any book, a screen will appear showing all the information related to it, meaning: title, author, publishing date, rating (with a starred ranking), category and abstract.

- **Scan Book:** when pressing this option, there will first be a verification that the user has the *Barcode Scanner* app installed on his smartphone. If not, he will be asked to install this app, and if this option is accepted, the user will be redirected to Google Play, where it will be possible to download it.

  Once the app is installed, if the user presses on this option, a screen will be accessed where a button allows to scan the book's barcode. When the book is recognised, thanks to its EAN13 code, the user will see a new screen that contains all the information related to the book. There will also be a question asking the user the wish to add such book to the spreadsheet, with two possible answers (Yes/No). If the user presses "Yes", all the information regarding the book will be saved on the sheet. If the user presses the option "No", the app will go back to the main screen.

- **Share:** by pressing on this option, the user will be shown a list of all the apps that allow the sharing of content, among the ones he has installed, such as Facebook, Twitter or Instagram.

- **Send:** this option accesses the apps that the user has installed on his smartphone for email sharing. In case the user only has one app of this kind, it will be accessed directly.

All the screens include a toolbar that shows, on the left side, the name of the current screen, and on the right side, an extended menu. By pressing on this menu, the user will have the possibility to access the options “Settings”, “About the app” and “Exit the app”. The app also includes language support, in this case Spanish, English and French. If the user changes the default language of his device, the app’s language will also change.

IV. Design and implementation

As mentioned before, the developed app for this project follows an architecture of the type client-server. The device, through this app, will make requests to a web server, in this case Google, which will attend the user’s request and give an answer through the Google Sheets API.

i. Implementation client architecture

The Android app that has been developed follows an outline that is basically made by Java classes, XML resources files (*layouts)* and the Android Manifest, each of which are stored in specific directories of the Android project.

The manifest, *AndroidManifest.xml,* is an XML file that must be included in all the Android apps, and that is its root itself. This file must specify:

- The application’s components and the relations among them.
- Support libraries (in addition to the Android’s ones).
- Required permissions for the execution of the app.
- Minimum API level required by the app.
- API Keys.

*All the elements included in this app’s Manifest are included in Table 6.

As for the types of resources that have been used in this project for the creation of the UI, this is a brief abstract of the ones implemented in this project:

- `Drawable`: this subdirectory includes the images, preferably under PNG format, that are used for the app’s design.

- `Layout`: it allows the definition of the distribution of the UI detached from the code. The most relevant are `LinearLayout`, `RelativeLayout and FrameLayout`.

- `Menu`: it allows the definition of the menus, both for activity and context, on XML files.

- `Mipmap`: folder for placing the app icon

- `Values:` among these are the XML related to strings, colours, dimensions and arrays.

- `Xml`: this subdirectory stores the arbitrary resources that can be consulted during execution, such as the app's settings, for instance.

The different external resources that have been created in this app, in addition to the images (stored in `Drawable`) and app icon (stored in `Mipmap`), can be seen on Table 7.

Finally, the Java classes created for this app are:

- **`About`**: shows the information about the developed app.

- **`Settings:`** class `PreferenceActivity` includes the settings.

- **`BookItem:`** class that contains a `BookItem` object, which represents a book, and comprises its cover and title.

- **`BookAdapter:`** adapter that acts as a bridge between the data obtained from the spreadsheet and the `ListView` where all the books are shown.

- **`GetBookInfo:`** this `Activity` works with the Google Books API and is intended only to obtain a specific book's information, which code EAN13 is passed by the `Activity ScanBook`.

- **`BookInfo:`** it retrieves the information of the book that the user has scanned (which data is received through the previous `Activity`), and show all their information on the screen. It also includes two buttons where the user can choose whether to add the book to his spreadsheet or not. If the user decides to save the book on his personal sheet, a call will be sent to the class `Spreadsheets`, which oversees the reading and writing on the spreadsheet. In case of pressing the "No" button, the user will be redirected to the main screen.

- **`MainActivity:`** it is the main screen of the app. It will be the first to be shown to the user when starting the app, and the one where redirected every time the button "Back" is pressed from any of the screens that the app has. It checks if it is the first time the user access to the app, and if so, it makes a call to the class `CreateSpreadsheets`. It is also in charge of writing on the `Navigation Drawer` the Gmail account that the user has chosen, and manage the change of the profile picture seen on top of it.

- **`My profile:`** it shows the user's profile information.

- **`ScanBook:`** it scans the books' barcodes. To access a book's information, the barcode format must be EAN13. The first time this screen is accessed, there will be a confirmation that the user owns the *Barcode Scanner* app. If not, there will be an option to access Google Play for its download. If the user doesn't install this app, there won't be any possibility to use this functionality.

- **`CreateSpreadsheets:`** creates the user's personal spreadsheet associated to the given Google account, and saves the sheet's identifier and the user's credentials, so that every time the user accesses the app, these are retrieved and a new registration/login isn't needed.

- **`Spreadsheets`**: class in charge of the reading and writing on the previously created sheet.

- **`AñadirFavorito:`** class that add a book as "Favorite" when the user presses the heart button on the `ListView` that contains all the books saved on the sheet.

Additionally, when pressing on the `Navigation Drawer's` options "Share" and "Send email", a list of the installed apps in the user's device will appear, depending on the option chosen. These two options are based on `Intents`; not related to `Activities`.

V. Conclusions and future work

After finishing the app development, it can be said that the main goals of the project, described on Chapter 1, are achieved successfully:

- The app creates a spreadsheet related with the user account. It can be accessed both from the smartphone and from Google Drive website.

- The app allows the user to write and read the data stored into the spreadsheet.

- The app allows the user to scan a book's barcode and retrieve the information about it.

- The app's interface is simple and intuitive.

Therefore, it has been possible to develop an access platform to a personal database stored in the cloud. This project has demonstrated the potential of using Google Sheets as a simple and intuitive alternative to the use of network structures. This way, small businesses can have the opportunity to collect their stock into a spreadsheet in a cheaper and easier way than the use of servers, and consult or modify it in real time in an efficient way.

i. Future work

After achieving the main goals of the project, it is a good idea to think about the possible future work in line with this project. The following ideas could be tested and implemented in a future:

- Adding the functionality of searching books by key words.
- Improvement of the user's interface.
- Improvement of the sharing functionality.
- Create a version of the app for iOs.
- Publish and distribute the developed app into Google Play.